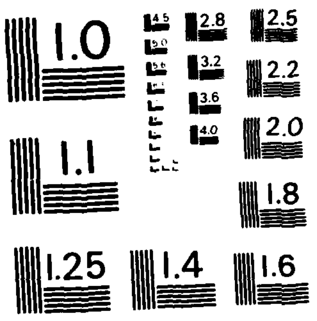

1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

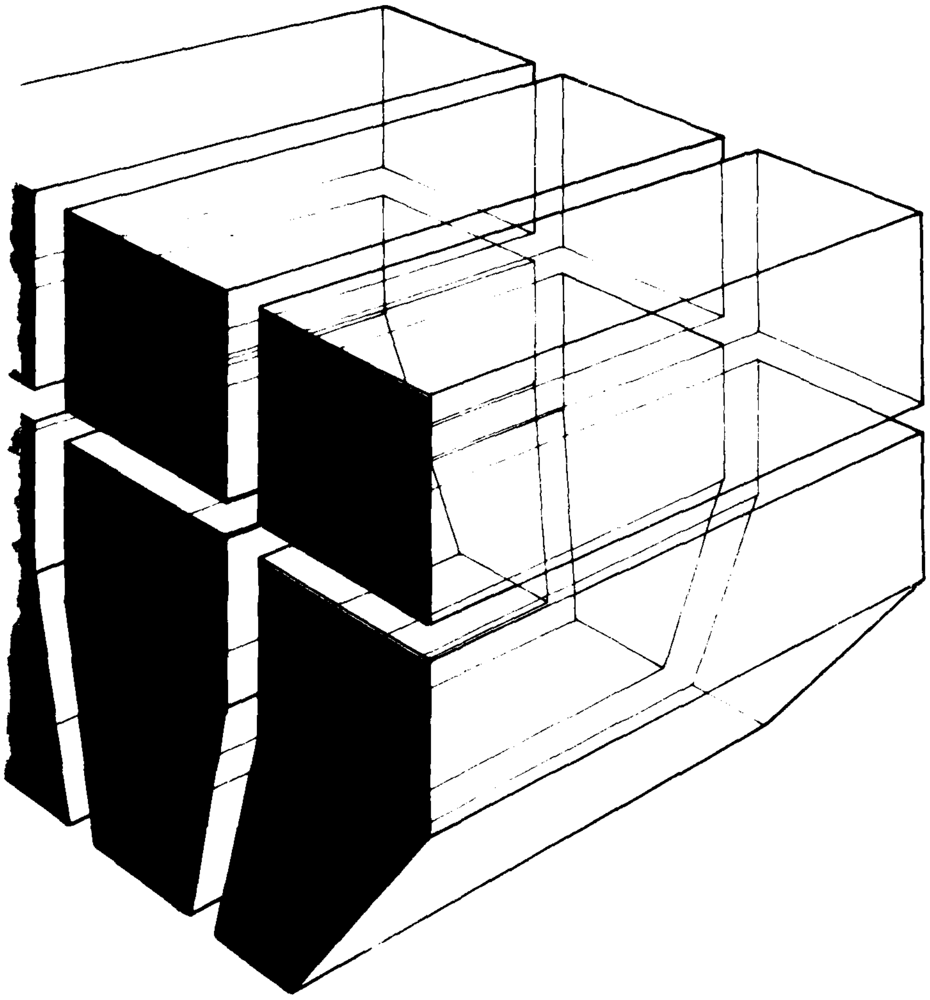
12

construction engineering research laboratory

United States Army Corps of Engineers
Serving the Army
Serving the Nation

TECHNICAL REPORT N-152
June 1983

USERS GUIDE: SIMULATION MODEL FOR AMMUNITION PLANTS; PREDICTION OF WASTEWATER CHARACTERISTICS AND IMPACT OF REUSE/RECYCLE



by
Steven Railsback
Manette Messenger
Ronald D. Webster
John T. Bandy

CERL

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CERL-TR-N-152	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Users Guide: Simulation Model for Ammunition Plants; Prediction of Wastewater Characteristics and Impact of Reuse/Recycle		5. TYPE OF REPORT & PERIOD COVERED FINAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Steven Railsback, Manette Messenger, Ronald D. Webster, John T. Bandy		8. CONTRACT OR GRANT NUMBER(s) IAO 81052
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Construction Engr Research Laboratory P.O. Box 4005 Champaign, IL 61820		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE June 1983
		13. NUMBER OF PAGES 65
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Copies are obtainable from National Technical Information Service
Springfield, VA 22151

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ammunition
manufacturing
water pollution
waste water
computerized simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the algorithm and details the operating instructions required for an ammunition plant process model developed for DARCOM environmental personnel. The model was created to define the impact of increased ammunition production on the quantity and quality of the effluents discharged from the plants. It also allows assessment of the impact of recycle/reuse of wastewaters on final effluent quality. This model may be accessed through the Environmental Technical Information System.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This work was performed for Headquarters, Army Materiel Development and Readiness Command (DARCOM) under IAO 81052 by the Environmental Division (EN) of the U.S. Army Construction Engineering Research Laboratory (CERL). The DARCOM Technical Monitor was Mr. Harry Delong.

Dr. R. K. Jain is Chief of EN. COL Louis J. Circeo is the Commander and Director of CERL, and Dr. L. R. Shaffer is Technical Director.

CONTENTS

FIGURES

USERS GUIDE: SIMULATION MODEL FOR AMMUNITION PLANTS; PREDICTION OF WASTEWATER CHARACTERISTICS AND IMPACT OF REUSE/RECYCLE

1 INTRODUCTION

Background

Army Ammunition Plants (AAP) manufacture explosives and propellants for use by all U.S. military services. The Army began to eliminate pollutant discharge from ammunition plants in the early 1970s. As a result, most AAPs now meet National Pollutant Discharge Elimination System (NPDES) discharge requirements; however, production is not even close to design capacity at any of the plants. Thus, it is not clear that wastewater treatment facilities could successfully treat a five- to ten-fold increase in pollutant loadings caused by capacity production to the levels required by NPDES permits. Therefore, the Army must identify the effects of increased levels of ammunition production on the ultimate amount of toxic pollutants discharged from AAP wastewater treatment facilities.

Objective

The objective of this study was to develop a computer modeling tool to help Headquarters, Army Materiel Development and Readiness Command (DARCOM) and AAP personnel predict: (1) the effluent quality of wastewater streams as a function of ammunition production level, and (2) the effect of various water reuse/recycle options of effluent quality. This report documents the model formulation, software development, and user instructions.

Approach

Needed model characteristics were defined, and a model was designed to satisfy these requirements (Chapter 2). Software and supporting documentation were then developed (Chapter 3).

2 MODEL FORMULATION

Defining Model Characteristics

The first step in designing a model that would meet Army requirements was defining its necessary qualities. Research indicated that a model combining the following characteristics would best meet these needs:

1. Ability to model any type of water use that may occur in AAPs, including water for nine manufacturing processes, washdown, cooling, and transporting finished ammunition through pipes.

2. Ability to specify a water-using process as many distinct steps or as a "black box."

3. Ability to change flow interconnections and re-run the model to assess the impact of recycle/reuse.

4. Creation of a user-friendly interactive environment to prompt the users for necessary data inputs and changes.

5. Use of the C programming language and UNIX operating system for all software developed in order to simplify eventual transport to DARCOM computers.

A model flexible enough to apply to the variety of AAP water-using activities allows the user to become familiar with one tool which can solve many problems. Designing such a "general-purpose" model means that very few assumptions are made about the characteristics of any particular process. Instead, the user must describe each process train fully.

General Description

The model designed to incorporate these defined characteristics includes four basic steps:

1. The user describes a process train.

2. The user describes water use, pollutant generation, and treatment efficiencies for each process in the train as a function of production level.

3. The program calculates water flows or pollutant loadings throughout the system, including the discharge; this is displayed immediately and saved in a file called RESULTS.

4. The user may change the system characteristics such as flow connections, treatment levels, etc., and re-run the program; this allows the analysis of various water reuse/recycling schemes.

The program describes the flow network with a set of simultaneous linear equations whose solution is the flow rates in each pipe. Each plant process, whether manufacturing or treatment, is referred to as a node. Nodes can also be pipe junctions or other situations where water or pollutant flow rates are changed. The flows between connected nodes are the variables the program solves for. Figures 1 and 2 are simplified flowcharts that describe how the plant model handles water flows and pollutant loadings.

The program describes the system of flows by means of the connection matrix -- the variable "array." The connection matrix is a two-dimensional array, with each node represented by a row and a column. The columns represent nodes which flow is "from," and the rows represent nodes which flow is "to"; a "1" in the matrix indicates connection between nodes. For instance, flow from node 3 to node 5 is represented by a "1" in column 3, row 5. The connection matrix allows the user to change flow pathways easily by simply inserting or deleting "1"s in the matrix. The routine called Matrxin creates the connection matrix.

Once the connection matrix is made, it is used by the routine Bildsys to generate the equations describing the system. These equations are represented by their determinant (variable "determ") and the vector of right-side values ("rhs"). (In matrix notation, the system of equations is defined by "determn[] [] * var[] = rha[] ".)

Several kinds of equations describe the inflows and outflows at each node, in a manner analogous to the use of Kirchoff's laws to solve electrical networks. The first set of equations, which is extracted directly from the connection matrix, simply states that the sum of inflows to a node equals the sum of outflows. When there is a node with more than one outflow, such as for a recycle, more equations are added to describe the fraction of the outflow which goes to each downstream node. At nodes where there are water inflows to the system, an equation is added which states that the inflows to the node, including inflows from the water source, equal the water demanded by the node. At this point, a variable representing the source flow at the node is added.

After all these equations are generated, there is one equation for each variable (pipeflow), and the equations can be solved. This is done by the computer with a Gaussian Elimination routine in the function Solvsys.

Following a more detailed description of each routine of the program, roughly in the order in which control normally flows. Figure 3 shows a simple process train which will be used to illustrate the results of each routine.

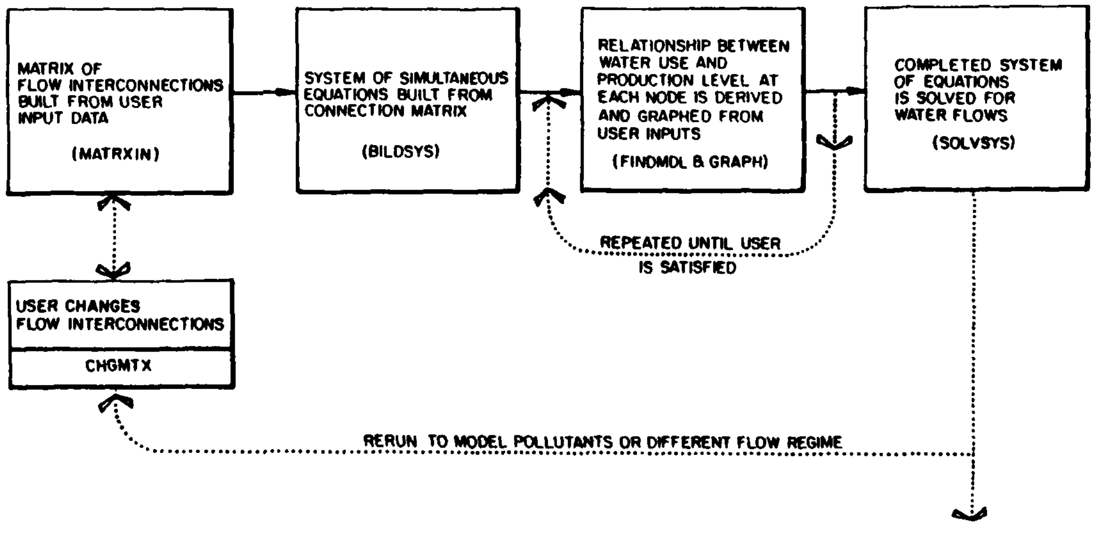

MATRIX OF FLOW INTERCONNECTIONS BUILT FROM USER INPUT DATA
(MATRXIN)
SYSTEM OF SIMULTANEOUS EQUATIONS BUILT FROM CONNECTION MATRIX
(BILDSYS)
RELATIONSHIP BETWEEN WATER USE AND PRODUCTION LEVEL AT EACH NODE IS DERIVED AND GRAPHED FROM USER INPUTS
(FINDMDL & GRAPH)
COMPLETED SYSTEM OF EQUATIONS IS SOLVED FOR WATER FLOWS
(SOLVSYS)
REPEATED UNTIL USER IS SATISFIED
USER CHANGES FLOW INTERCONNECTIONS
CHGMTX
RERUN TO MODEL POLLUTANTS OR DIFFERENT FLOW REGIME

Figure 1. Simplified flowchart of hydraulic model.

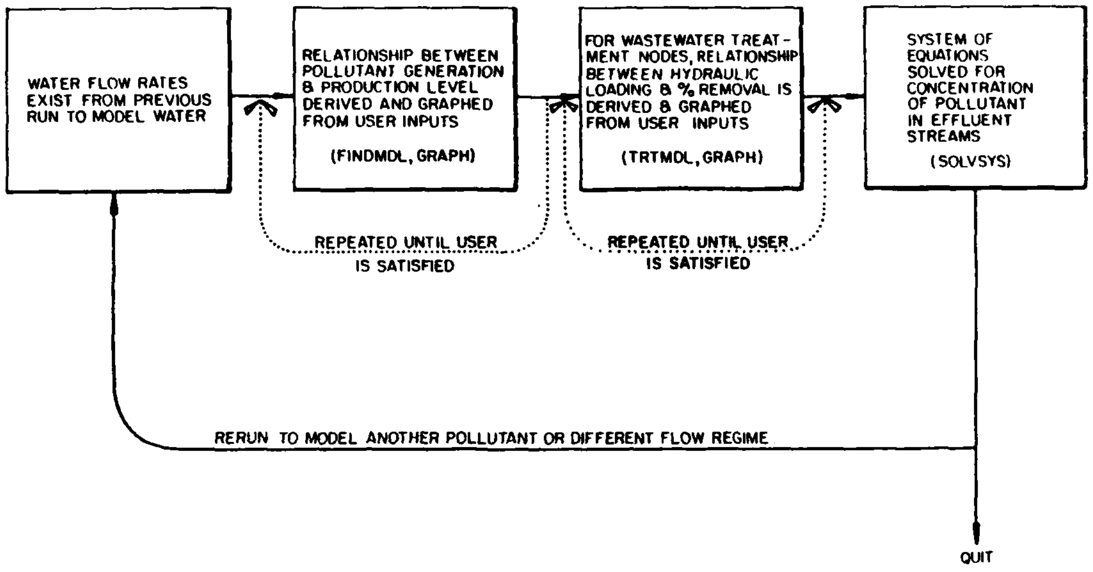

WATER FLOW RATES EXIST FROM PREVIOUS RUN TO MODEL WATER
RELATIONSHIP BETWEEN POLLUTANT GENERATION & PRODUCTION LEVEL DERIVED AND GRAPHED FROM USER INPUTS
(FINDMDL, GRAPH)
FOR WASTEWATER TREATMENT NODES, RELATIONSHIP BETWEEN HYDRAULIC LOADING & % REMOVAL IS DERIVED & GRAPHED FROM USER INPUTS
(TRTMDL, GRAPH)
SYSTEM OF EQUATIONS SOLVED FOR CONCENTRATION OF POLLUTANT IN EFFLUENT STREAMS
(SOLVSYS)
REPEATED UNTIL USER IS SATISFIED
REPEATED UNTIL USER IS SATISFIED
RERUN TO MODEL ANOTHER POLLUTANT OR DIFFERENT FLOW REGIME
QUIT

Figure 2. Simplified flowchart of pollutant model.

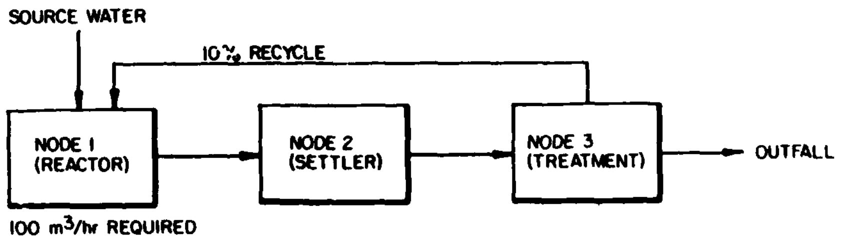


Figure 3. Example process train.

Routines

Matrxin

Matrxin builds the connection matrix from interactive input. It was designed to minimize input effort by allowing the user to describe nodes with alphanumeric names of 20 or fewer characters. The user is asked to put in the name of a node and the name of the nodes(s) to which flow goes. This is done for all nodes; the program builds the matrix from this information. The information for each node (its name and the names of the nodes flow goes to) is stored in a structure called "node." "node.ndname" is a character string with the node's name, and "node.ndto[]" is an array of character strings containing the names of the nodes flow goes to. The program is presently dimensioned for up to 29 nodes and up to five outflows for each node. (The structure "node" is dimensioned to 30, but the 0 element is not used; likewise, the character arrays tonode and flofrac are dimensioned to 6 but only 5 are used.)

Flows to the outfall (or "sink") should not be put in as nodes. Matrxin asks the user which nodes go to the outfall and automatically includes these flows in the flow network. Likewise, flows into the network from the source, whether it is a river or water treatment facility, are automatically included for any node at which there is a water demand (this is done in routine Bildsys).

To avoid having to enter all the node information every time the program is run, an option has been added to allow the user to read all the input values for Matrxin from a file. Matrxin returns a value to the main routine which is the size of the connection matrix which is also the number of nodes in the system. Figure 4 shows an example connection matrix.

Chgmtx

The routine Chgmtx allows the user to modify the connection matrix either to correct input mistakes or to model different flow routes. The routine asks the user which node to change; this refers to changing the outflows. The nodes to which flow is going are printed out, and the user is asked which nodes, if any, to delete. The user may then add flow to any existing node.

This routine may be called immediately after the input routine or before starting a subsequent run of the program.

Bildsys

Bildsys builds the system of simultaneous equations. Because it is also the main function, it is known in the program as "main." After calling functions Matrxin and Chgmtx, Bildsys uses the connection matrix to write the first group of equations.

The variables are labeled by the program before the equations are generated. Since each "1" in the connection matrix represents a flow, there will be one variable for each "1." To keep track of them, the program goes through the matrix from row 1, column 1, moving across each row and numbering the "1"s encountered in ascending order. These numbers are then referred to as "variable numbers." Since the "1" in the matrix is replaced by its variable number, the variable number for a flow is expressed as "matrix[row] [col]." There will be a row and column in the determinant for each variable number. Figure 5 shows a numbered connection matrix.

The first group of equations (there will be one for each node), simply states that the sum of inflows equals the sum of outflows at each node. This is done

	NODE 1	NODE 2	NODE 3
NODE 1	0	0	1
NODE 2	1	0	0
NODE 3	0	1	0
OUTFALL	0	0	1

Figure 4. Connection matrix built by routine Matrxin.

	NODE 1	NODE 2	NODE 3
NODE 1	0	0	1
NODE 2	2	0	0
NODE 3	0	3	0
OUTFALL	0	0	4

Figure 5. Numbered connection matrix (variable numbers) in routine Bildsys.

by moving through the connection matrix and at each variable (element in the matrix with a non-zero value), putting a +1 in the determinant at row = row in matrix, column = variable number, and a −1 at row = column in matrix, column = variable number. In other words, the program looks at a flow, finds the node at which the flow is an inflow, and puts a +1 in the equation for that node. It then finds the node for which it is an outflow and puts in a −1 in its equation. For the bottom row of the connection matrix, which is for the outfall, only −1's are added, since flow does not go to anywhere from the outfall. The result is one equation for each node. Within the determinant, the row number is equal to the number of the node the equation is for; the column numbers correspond to variable numbers, with one additional column for flow to the outfall.

Bildsys then asks the user which nodes are processes at which there is a water demand. These are nodes at which the process being modeled requires a certain amount of water; this amount is determined from the production level by the function "findmdl" (explained below). For such nodes, an additional variable must be added to represent the amount of water coming from the source to this node. An equation is added which states that the sum of inflows to the node, including the source flow, equals the water demand as determined by Findmdl. The model assumes that if there is a water demand at a node and if there is insufficient inflow from upstream nodes to meet the demand, water from the source is added.

The equation is added by making a new row in the determinant which has a +1 in each column corresponding to a variable number in the node's row of the connection matrix. The inflow is then added by putting a +1 in a new column, which then represents the inflow; the required flow rate is added to the right-hand side vector (rhs) for this new row.

Because another inflow has been added to the node, each time this procedure is used to add source flow, the program must go back and add a +1 to the node's "inflow = outflow" equation. This is done by first looking in the node's column in the connection matrix and finding the first variable number in the column. In any row of the determinant in which there is a −1 in the column equal to that variable number, a +1 is added to the column corresponding to the inflow.

The final group of equations added describes how flow is divided at nodes with more than one outflow. This is necessary to provide enough equations to solve the system, since an extra outflow means an extra variable. The equations needed at each node include one for each extra outflow, or one less than the number of outflows. It is assumed that a constant fraction of the total outflow will go to each downstream node.

The program moves through the columns of the connection matrix. Any column with a non-zero value in more than one row has more than one outflow. At this point, the user is asked for the fraction of the outflow going to one of the downstream nodes. This question is repeated so that all the equations needed can be written. Each equation states that (1 - flofrac[i]) times the outflow to node i plus (−1 * flofrac[i]) times all the other outflows equals zero (flofrac[i]) equals the fraction of the outflow to node i). This equation divides the flow between node i and the other outflow nodes. Figure 6 is an example of a determinant built by Bildsys.

Modeling consumptive use of water is not now included in the program, but doing so would not be difficult. Water consumption either as a constant rate, as a constant fraction of water use, or as a function of production level can be modeled by altering the "inflow = outflow" equations. This process is similar to the way pollutant removal is handled, as described on p 13.

At this point, the set of equations is complete and Bildsys calls function Solvsys to solve them. The above description of Bildsys has assumed that water flows are being modeled and there are some differences when pollutants are being modeled instead. Instead of asking if a water demand function exists, the program asks if there is pollutant generation at each node. If the answer is "yes," function Findmdl determines the amount of pollutant added to the system instead of the water demanded. Instead of adding a variable and an equation describing water inflow, Bildsys modifies the existing "inflow equals outflow" equation to say "inflow plus pollutant generated equals outflow." This is done by changing the right side of the equation from zero to the amount of the pollutant generated. If no pollutant is generated at a node, the program asks the user if the node is a waste treatment process where the pollutant is removed. If it is, the program uses the function Trtmdl to determine the percent removal of the pollutant. The "inflow equals outflow" equation is then modified to "inflow times removal equals outflow." This is done by multiplying the +1 (inflow) values in the equation by the removal fraction.

VARIABLE NUMBER

	1	2	3	4 (FLOW TO OUTFALL)	5 (INFLOW AT NODE 1)		RHS	MEANING
EQ. # 1	+1	-1	0	0	+1	=	0	INFLOW = OUTFLOW AT NODE 1
# 2	0	+1	-1	0	0	=	0	INFLOW = OUTFLOW AT NODE 2
# 3	-1	0	+1	-1	0	=	0	INFLOW = OUTFLOW AT NODE 3
# 4	0	+1	0	0	0	=	100	OUTFLOW AT NODE 1 = 100 (FLOW REQUIRED)
# 5	0.9	0	0	-0.1	0	=	0	RECYCLE 10% OF NODE 3 OUTFLOW

SOLUTION: VAR # 1 = 10
# 2 = 100
3 = 100
4 = 90
# 5 = 90

Figure 6. Determinant (system of equations) built by routine Bildsys.

The final section of Bildsys prints out the various flows after the system of equations has been solved and then allows another run of the program.

Findmdl

The routine Findmdl returns the amount of water demanded or pollutant generated at a node when called from Bildsys. This is done when the program searches a file of existing process models, which are two-dimensional graphs of production level (as percent of full production) versus water demanded or pollutant generated. The name of each file is the name of the constituent.

The program first checks whether such a file exists by using the UNIX library function "open." If "open" returns a negative value, the file does not exist and one is created with the library function "creat." If the file does exist, it is searched for title lines — lines which begin with the character "x." If the title is the same as the node name in question, the model is read. If no model is found with the node's name, a new model must be put in. Thus, the user should use the same node names to use existing process models.

If an existing model is found, it is graphed out by the function Graph, and the user may replace the model. If the user does not want to change the existing model, it interpolates the desired water demand or pollutant generated and returns it. In all other cases (a new file created, no existing model found, or existing model to be replaced), a new model is generated and written into the file, overwriting the existing model if there was one. The user is asked to put in the production level and corresponding water demand or pollutant generated for arbitrary number of points, up to 20. When a production level of 100 is entered, the routine automatically stops, calls the graphing routine, and asks if the model should be changed. When the user is satisfied with the model, Findmdl interpolates the water demand or pollutant generation from the production level specified by the user.

An important assumption built into Findmdl (and Trtmdl) is that when a production level is modeled which is less than the first point in the model, the water demand is the same as in the first point of the model. In other words, the program does not assume that water demand goes to zero as production goes to zero. This assumption could be changed with a minor program modification in the interpolation routine.

Trtmdl

Trtmdl is very similar to Findmdl; the difference is that Trtmdl is used to handle modeling of wastewater treatment processes, while Findmdl is used to handle pollutant generation. Instead of modeling water demand or pollutant generated vs. production level, it models removal efficiency vs. flow rate, a function which is more traditionally used in wastewater engineering and modeling. Once the proper model is found or put in, exactly as in Findmdl, a value of percent removal is returned. To do this, a value of the flow rate must be obtained. This value is stored in the structure "nodes" under the variable "watrin," where its value was assigned the last time the program was run for water flows. This means it is important to model water flows before modeling pollutant flows and removal.

Graph

The function Graph is a simple graphing routine which uses two one-dimensional arrays as input. It generates 50 points between zero and the maximum value of the x=axis, then plots them out on 20 rows of 50 columns each.

Solvsys

Solvsys solves the system of linear equations generated by Bildsys, using Gaussian Elimination on the determinant "determ" and the vector "rhs" of right-side constants. Gaussian Elimination requires that the elements of the main diagonal of the determinant be non-zero to avoid division by zero.

Solvsys uses two separate methods to insure non-zero values in the diagonal. First it moves down the rows and if it finds a row with a zero on the diagonal, it looks below that row for another row with a non-zero value in the desired column. If it finds such a row, the rows are interchanged. After going through the entire determinant in this fashion, most (but not necessarily all) of the diagonal elements are non-zero and the Gaussian Elimination routine starts.

The routine starts at row 1, column 1, and looks down the rows until it finds a non-zero value in column 1. When this happens, row 1 is multiplied by the appropriate value and added to this row to make the value go to zero. The program proceeds in this fashion down through all the columns for each row. If a non-zero element is found in a column whose diagonal element is still zero, the element's row is added to the diagonal's row to get a non-zero diagonal. When this process is finished, all values below the diagonal of the determinant are zero; Solvsys then goes back, starting with the last variable, and evaluates all the variables. A variable is evaluated by using the row with it on the diagonal (the row number equal to the variable's column number); its value is the right side of the row, minus the sum of all other variables to the right of the variable times their coefficients in the determinant; this number is divided by the coefficient of the variable in question. The values of the variables are stored in the array "var[]." At this point, control returns to the main routine Bildsys, which prints out the results and writes them into a file called "RESULTS."

Help

Help is called whenever the user replies to a prompt with the word "help." Help includes some general instructions on using the program and specific instructions for each prompt. The instructions for each prompt are based on the information in the user's guide (Chapter 3).

Model Verification and Field Application

Enough data has not yet been obtained from any single ammunition manufacturing process to do a verification study of the model. There are many publications from AAP water pollution studies, but there is no complete set of information on process descriptions, water use, pollutant generation, wastewater treatment capacities, and effluent characteristics, all measured under the same production conditions; such data is needed to make a realistic model run. The problem is compounded because much wastewater treatment modernization, including water reuse and recycling, has been done since plants stopped large-scale production; in fact, the present lack of activity at AAPs will hamper any modeling effort.

Conversations with Government engineers at AAPs have determined that the input data needed to run this model is available, though it may have to be extracted from operating logs. Unfortunately, these logs are not available to non-DARCOM personnel, which indicates that the program should be used by plant operators or overseers, since they are most familiar with plant operations and production levels.

One problem which will be encountered when the model is used in the field is that some production lines do not really have gradations of production level; i.e., they are either on or off. The model can handle this situation in several ways. If, for instance, a production line is either on or off during each of three daily shifts, the water use/pollutant generation graphs can be drawn as three-stage step functions: for up to 33 percent production, the pollution generated by 8 hours (one shift) of continuous operation is used; for up to 67 percent production, the pollution generated by 16 hours (two shifts) is used; and for 100 percent production, the pollution generated by 24 hours (three shifts) is used. In such cases, water flows will be averaged over the whole day, which is realistic if there is much detention time in the wastewater treatment system.

The program can also be used with flat water use/pollutant generation curves, meaning that pollutant generation is independent of production level, or that production level is irrelevant. In these cases, where the production line is either on or off, the effluent is readily predicted by using historical measurements of pollutant flow as the (flat) pollutant generation graph. When the line is on, these measured flows will be routed into the treatment models to predict the effluent; when the line is off, no effluent will be predicted.

Another problem in field use will be that in some cases, data from only one production level will be available. In this case, the user will have to estimate the shape of the curves; usually this will mean extrapolating toward higher production levels. An experienced ammunition plant operator should know to some extent how water use and pollution generation will change as production increases; for instance, the operator may feel that the amount of washwater will remain constant if washdown operations are not increased with production, while the discharge of some pollutant from the actual production line may be estimated to be linearly related to production level. This requirement for user judgment is another reason why plant operators should use the program.

Possible Model Modifications

The plant model could be further developed to include:

1. Built-in models of the manufacture of explosives and propellants other than TNT. Currently, such models must be input by the user.

2. A method to account for the consumptive use of water and reaction kinetics of pollutants. A kinetic model would involve a major effort; however, a model for consumptive use of water could be added easily.

3. A checking routine for process trains involving reuse/recycle to compare the recycled water quality against the water quality needed for process influent.

4. Environmental fate of toxic pollutants in the receiving stream. A number of adequate fate models exist, but they are not written in the C programming language.

5. Costing algorithms and expected treatment efficiencies for various treatment technologies used to treat ammunition plant process wastes.

3 USERS GUIDE

The plant model is available as Experimental Profile Number 30 of the Environmental Technical Information System (ETIS).[1] ETIS can be accessed over commercial telephone lines by almost any kind of computer terminal. ETIS is an interactive, answer-driven umbrella system which can be used by people unfamiliar with computers.

Accessing ETIS

If you decide to access the plant model after interactively entering ETIS, you need only transfer into the program. The following paragraphs provide instructions for accessing ETIS information by remote terminal.

Once you have acquired a log-in and a password from CERL's Environmental Division, you can access ETIS by remote terminal by following the directions in CERL Technical Reports N-56[2] and N-2[3] (DA Pamphlet 200-2[4]), and CERL Technical Report

[1] R. D. Webster, et al., *Development of the Environmental Technical Information System*, Interim Report E-52/ADA 009668 (U.S. Army Construction Engineering Research Laboratory [CERL], 1975).

[2] J. van Weringh, J. Patzer, R. Welsh, and R. Webster, *Computer-Aided Environmental Legislative Data System (CELDS) User Manual*, Technical Report N-56/ADA061126 (CERL, 1978).

[3] R. D. Webster, R. A. Mitchell, R. L. Welsh, E. Shannon, and M. L. Anderson, *The Economic Impact Forecast System - Description and User Instructions*. Technical Report N-2/ADA027139 (CERL, 1976).

[4] *The Economic Impact Forecast System–Description and User Instructions*, DA PAM 200-2 (Department of the Army, December 1976).

N-43.[5] After acquiring a remote terminal and a telephone, dial the system's number (FTS 217/333-4086, WATS 800/637-0958). If there is no answer, the entire system is down for maintenance. Upon hearing a steady tone, plug the phone into the terminal, making sure that the earpiece and the speaker are in the proper openings, and log into the system. After logging in with the correct name and password, you will receive system messages. If the system indicated "NO DIRECTORY" or a similar designation, access to ETIS is probably closed down and you should try again later. If the system is in operation, output similar to that shown in Figure 7 will appear on the screen or terminal. (The text on the right-hand side of the figure is explanatory and is not part of the output.)

Keep several things in mind when using the system. The symbol <CR> used in some instructions means to depress the carriage return button. The instruction to type CTRL-d means to simultaneously depress the button marked CTRL and the letter d. An input error can be corrected by typing CTRL-h (hitting the CTRL and h buttons simultaneously) if the return button has not yet been depressed. This procedure will back the carriage up one space each time it is repeated. This can be done as many times as necessary. Every symbol which has been backspaced over has been removed from the terminal memory. Therefore, if the first digit of a six-digit number has been mistyped, you must depress CTRL-h six times and then retype all six digits. The corrected symbols will be overprinted on the paper. To stop a long listing depress the button marked DEL (delete).

The Plant Model

Like ETIS, the plant model also contains an answer-driven system to accept required input data from the user. Five types of information are required to run the model:

1. A process train showing all flow interconnections, source water inputs, and outfalls.

2. Volume of source water necessary (cubic meters/hour or cubic meters/batch for batch processes) at a minimum of two different production levels (percent of full production).

3. Volume of each pollutant (constituent) generated (kg/hour or kg/batch) in the water phase versus the production level (percent of full production).

4. Efficiency (percent removal) of all treatment processes for each pollutant (constituent) with respect to flow rate through treatment process (cubic meters/hour or cubic meters/batch).

5. How the process train could be modified to incorporate recycle/reuse.

The model returns values for the flow rate and pollutant loading for each interconnection in the process train, including the outfalls. Then modification of the existing process train or specification of a new one is allowed before the program is rerun.

Several general instructions should be followed throughout the program.

1. All process names are alphanumeric words which can contain any character, but should not include blanks and must be no more than 20 characters long. It is important to maintain spelling and upper/lower-case consistency. It is also important to make sure that a process' name is the same as the name of the process model, if any, that is on file. At some, but not all prompts, the program will find a spelling error and let the user correct it.

2. Yes or no questions can be answered by spelling out yes, no, or just by y or n. Upper/lower case is not important.

3. The first constituent that should be modeled in any run is water because water flows are needed by the treatment process models. Changes in flow routing or in production level change water flow rates, and the program reminds you to model water again after making these changes.

4. At any point in the program where a list is typed in, the program will keep repeating the same prompt. Typing a 0 (zero) tells the program that the user is through entering the list and wants to move on to the next prompt. In general, "0" acts as an escape.

5. At any prompt which is followed by "h/a," on-line help is available. Simply answer the question with the word "help" to obtain a help routine which provides information similar to that given in this report.

[5] S. E. Thomas, R. A. Mitchell, R. E. Riggins, J. J. Fittipaldi, and E. W. Novak, *Computer-Aided Environmental Impact Analysis for Industrial, Procurement, and Research, Development, Test, and Evaluation Activities–User Manual*, Technical Report N-43/ADA056997 (CERL, 1978).

```
                    ETIS (Trademark applied for)

                United States Army Corps of Engineers
              Environmental Technical Information System


ETIS: What program? (Type <cr> to see list):
Type:
  1 or intro          for an introduction to the Environmental Technical
                         Information System and people to call for help with
                         problems about using the programs.
  2 or eics           for the Environmental Impact Computer-aided System.
  3 or celds          for the Computer-aided Environmental Legislative Data System.
  4 or eifs           for the Economic Impact Forecasting System.
  5 or afeics         for the Air Force Environmental Impact Computer-aided System.
  6 or iicep          for the Interagency/Intergovernmental Coordination for
                         Environmental Planning (IICEP).
  7 or xper           for the Experimental Subsystem Module of ETIS -
                         Pilot Systems under development.
  8 or help           for HELP on using any of the ETIS systems or UNIX.
  9 or rubouts        to ignore extraneous phone noise.
 10 or end or bye     to exit from ETIS.
  ! mail              to see your mail.
```

Figure 7. Example ETIS instructions and explanation.

Step-by-step instructions for using the model are given below, in the order in which the program generally flows. Because you can choose how the program proceeds at several points, some prompts may be repeated during a session.

1. Do you want process configuration input to come from a file? (h/a)

This allows you the option of keeping the answers to questions 3 through 5 (below) in a file. Be sure to include all the zeroes in the file to terminate questioning (see general instruction 4 above).

2. What is the name of the file? (h/a)

Question 2 asks the name of the input file if question 1 was answered affirmatively.

If question 1 was answered negatively, questions 3 through 5 will be asked.

3. What is the name of the next node? (h/a)

The first step in the program is to describe the flow network. This is done by first typing the names of each node, in any order.

4. What node does flow go to from <nodename>? (h/a)

For the node just entered in response to question 3, enter the name(s) of the node(s) to which flow goes. When there are no more, type in "0."

Questions 3 and 4 will be repeated until all nodes are entered; at this point, type in "0" in response to prompt 3.

5. Enter node which goes to sink. (h/a)

Type in the name of any process from which flow goes to the outfall. This prompt will be repeated, allowing multiple outfalls, so type "0" after the last one.

At this point, the flow connections are printed out.

6. Do you want to change flow connections? (h/a)

To change where flow goes for any of the nodes, answer yes. This question is repeated at the beginning of each run, so new flow routing schemes can be tried during the same session.

Prompts 7 through 9 occur if flow connections are changed.

7. Which node do you want to change? (h/a)

Enter the name of the node whose outflow you wish to re-route. This instruction will be repeated at the end of the change routine, so you can change as many nodes as desired.

At this point, the nodes which flow goes to from the node you want to change are printed out.

8. Which node do you wish to eliminate flow to? (h/a)

If you want to stop flow from going to a node, enter the node's name. You can eliminate flow to the outfall by entering "outfall." This question will be repeated so you can eliminate flow to more then one node, if necessary.

9. Which node do you wish to add flow to? (h/a)

If you want to direct flow to a node, enter the node's name. If not, enter "0." You can add flow to the outfall by entering "outfall." This question is also repeated.

After all changes have been made, or if no changes were requested, the following prompts are given.

10. What production level? (h/a)

Enter the production level you want to model. This should be expressed as a percent of full production, so it should be between 0 and 100.

11. What constituent do you want to model? (h/a)

At this point, the program also prints out a list of the constituents for which models exist, and you should choose from them. You may also add a new constituent to the list by entering its name. For the first run, and after any changes in flow routing or production level, water should be modeled first. The prompts now given depend on whether water or a pollutant is being modeled. The prompts for modeling water flows are explained in questions 13 through 15. The prompts for modeling pollutant generation are explained in questions 19 through 28.

12. Do you want to use the existing <constituent> models? (h/a)

If you want to use the models already on the file, without any modifications and without seeing the models, answer "yes." This will bypass prompts 13 through 16, 20 through 23, and 25 through 28.

13. Is there a water demand at <nodename>? (h/a)

If the process represented by the nodename is one which requires a certain amount of water, answer "yes." This tells the program that you wish to model the water demand at this node.

If question 13 is answered "yes," the program looks for a file of existing models which includes this node. If the file is not found, or if no model with the node's name is found in the file, the next prompt will be question 15. If a model is found, it will be graphed out on the terminal.

14. Do you want to change this model?

If the graph adequately represents the water demand vs. production level for this node, answer "no." Otherwise, answer "yes," and you will be able to type in a new model by answering questions 15 and 16.

15. Production level?

16. Water?

Each tin e these two prompts are given, put in a point on the water demand vs. regular production level graph. You need not enter a point for zero production. When a production level of 100 is entered, the program automatically moves on.

The model is now graphed ou., and you will be asked again if you want to change it. When you are satisfied with the model, answer "no" and the new model will be written into the file.

Questions 13 through 16 will be repeated for all nodes.

17. Enter fraction of flow from <nodename> to <nodename>. (h/a)

Wherever there is a divided outflow from a node, the program must know what fraction of the flow goes to each outflow. It will ask this question for all nodes which have divided outflows. The value entered should be between zero and one.

After question 17 has been answered, the program has all the input it needs and will compute the flows between all nodes. The results will be typed out at the terminal as well as appended to a file called RESULTS, which can be accessed after the session is over.

18. Do you want another run? (h/a)

Answering "no" will terminate the session. "Yes" will take you back to question 6.

The prompts explained below are for modeling pollutant flows. The following sequence will follow question 11 if a pollutant constituent is chosen for modeling.

19. Is <constituent> generated at <nodename>? (h/a)

If the node is one at which the constituent enters the wastewater stream, answer "yes," and pollutant generation will be modeled, as explained in questions 20 through 23. A "no" reply will lead to further prompts, as explained in questions 24 through 28. Replying with an "x" tells the program that this node is not a treatment process; this bypasses prompts 24 through 28.

After a "yes" reply to question 19, the program searches for a model of pollutant generation vs. production level in a file. If no file exists or if there is no model in the file, you will have to build one as in questions 21 and 22. If a file does exist, it will be graphed out and question 20 asked.

20. Do you want to change this model?

If you are satisfied with the existing model, answer "no." Otherwise, questions 21 and 22 will be used to build a new model.

21. Production level?

22. <constituent>?

Put in the points for a graph for the amount of the pollutant constituent generated vs. production level. Production level should be expressed as a percent of full production, i.e., between 0 and 100. You need not put in a point for zero production; the program stops asking for points after you have entered a production level of 100.

23. Do you want to change this model?

If you are satisfied with the new model which has just been graphed, answer "no," and it will be written into the file. Otherwise, you will go back to question 20.

If the node was not one where the pollutant was generated, the following series of prompts will appear.

24. Is <nodename> a treatment process where <constituent> is removed? (h/a)

Answer "yes" if you want to model the removal of the pollutant at this node, and you will get prompts 25 through 28. A "no" moves you on to the next node.

Modeling treatment processes is very similar to modeling generation or water demand, except you are modeling percent removal vs. flow rate for treatment processes.

At this point, when modeling a treatment process, the program looks for a file containing a model for the process. If no file is found or no model with the node's name is found in the file, questions 26 through 28 will be used to build a model. If a model is found, it will be graphed out and question 25 asked.

25. Do you want to change this model?

If you are satisfied with the existing model of treatment efficiency (y-axis) vs. inflow rate (x-axis), answer "no," and the program will use the model. Otherwise, answer "yes," and you will get questions 26 through 28 to build a new model. If your model does not include flow rates as high as you are trying to model, you will have to build a new model.

26. Flow rate?

27. Removal?

For up to 20 points, put in flow rate vs. removal. Type in "0" for flow rate to end the input and move on. The new model will be graphed for you at the terminal.

28. Do you want to change this model?

If you are satisfied with the model you just built, answer "no," and the program will return to question 19 until all nodes have been examined. A "yes" reply will return you to question 26.

Appendix A provides an example session of the model, and Appendix B provides the program listing.

4 SUMMARY AND RECOMMENDATIONS

A pilot ammunition plant computer-aided simulation model was designed and programmed to predict wastewater effluent quality at various production levels and to assess the impact of reuse/recycle on effluent quality. Any type of process train can be used as input to the model to describe water use in AAPs. The plant model may be accessed through the Environmental Technical Information System.

Detailed model input data on wastewater quality and quantity for cooling, washing, and transport operations is not available in the literature. Such information is probably available in AAP operating logs, but would take some time to extract, and it may not be available for more than one production level. Appropriate estimations and assumptions can be made by the user when water use and pollutant generation are not well-defined functions of production rate.

It is recommended that the plant model described here be tested and used by AAP and other DARCOM personnel before being further developed or refined.

APPENDIX A: EXAMPLE SESSION

This example run is for a simplified version of the sellite wash process for TNT, the source of so-called "red water." For this example, a pond is the only treatment process. In recent years, "red water" has either been sold for paper processing or dried and incinerated. In the example, a hypothetical recycle of the pond effluent is considered.

The model's level of resolution can be chosen by the user; nodes may be chosen for units as small as individual reactors, or whole process trains may be included in one "black box" node. A degree of resolution should be chosen that is suitable for the data available.

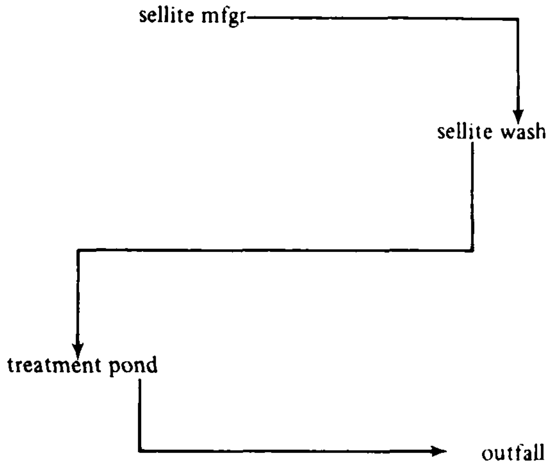
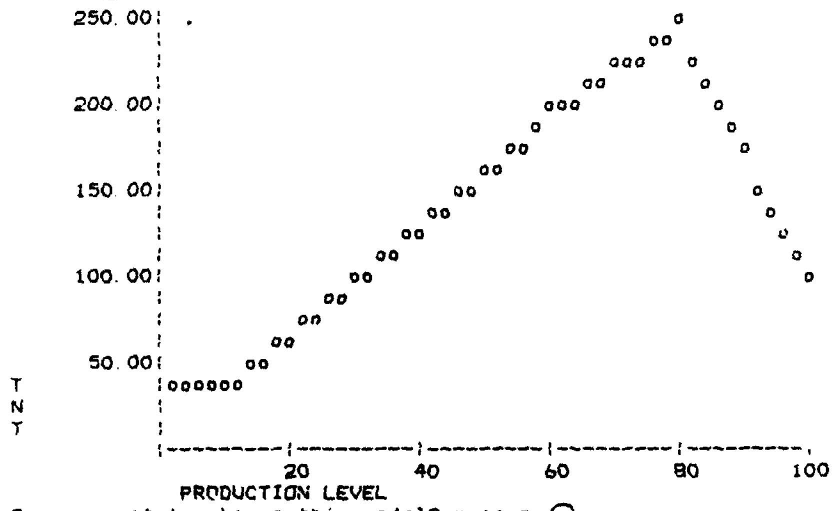


Figure A1. Example process train.

```
  Plant Model Program

 At all prompts which are followed by: (h/a) you can get on-line
help by responding with the word 'help'.

Answer yes-no questions with yes, no, y, or n



Do you want process configuration input to come from a file? (h/a) help

Do you want general help or help specific to the question you are on?
Enter 1 or 2:

    1= general
    2=specific
1


Ammunition Plant Process Model

  This is a program to model the flow of water and water-borne pollutants
The following general instructions apply:

1. All process (node) names should contain 20 or fewer characters of
any type, but without blanks.  Be careful with spelling!

2. Yes or no questions can be answered with: yes, no, y, or n.

3. Whenever a prompt repeats itself you can tell it to move on
by typing '0' (zero).

4. You must always model water flows before pollutant flows because treatment
models use flow rate as a parameter.  Any time you change the production
level or re-route flows you should model water flows again.

5. At any prompt where '(h/a)' appears, you can answer 'help' and get it.

More details are available in the user manual and details are available
here for specific prompts.

Hit '1' to continue
1
Choose one of the following:


   1  Do you want input to come from a file?
   2  What is the name of the file?
   3  What is the name of the next node?
   4  What node does flow go to?
   5  Enter node which goes to outfall
   6  Do you want to change flow connections?
   7  Which node do you want to change?
   8  Which node do you wish to eliminate flow to?
   9  Which node do you wish to add flow to?
  10  What production level?
  11  What constituent do you want to model?
  12  Do you want to use the existing models?
  13  Is there a water demand at this node?
  15  for help with units.
  17  Enter fraction of flow from a to b
  18  Do you want another run?
  19  Is the constituent generated here?
  24  Is this a treatment process where constituent is removed?
  98  for intrepretation of results
  99  to quit HELP and return to program

Which one do you want? (you entered 'help' from no. 1) 1

Help for question 1:
Do you want process configuration input to come from a file?

If you have a file with all the node names, where flow
goes to for each one, and the names of nodes which drain
to the outfall, you can use it instead of answering
questions 3 through 5.  This is very useful when running a
model several times.  If you answer yes, the next question
will ask you the name of the input file.

More help?  y or n n

Do you want process configuration input to come from a file? (h/a) n

Process configuration entry routine: put in the name
and where flow goes to for each node.


What is the name of the next node?   Hit zero for no more  (h/a) sellitemfr
sellitemfr
What node does flow go to from sellitemfr? (h/a)  sellitewash
sellitewash
What node does flow go to from sellitemfr? (h/a)  0
0
What is the name of the next node?   Hit zero for no more  (h/a) sellitewash
sellitewash
What node does flow go to from sellitewash? (h/a)  treatmentpond
treatmentpond
What node does flow go to from sellitewash? (h/a)  0
0
What is the name of the next node?   Hit zero for no more  (h/a) treatmentpond
treatmentpond
What node does flow go to from treatmentpond? (h/a)  0
0
What is the name of the next node?   Hit zero for no more  (h/a) 0
0
Enter node which goes to sink (h/a) treatmentpond

Enter node which goes to sink (h/a) 0

Flow goes from sellitemfr to sellitewash

Flow goes from sellitewash to treatmentpond


Flow goes from treatmentpond to sink

Do you want to change flow connections? y or n (h/a)  n

What production level- 0 to 100? (h/a) 100

What constituent do you want to model?

The following are available: (h/a)

water
```

```
TNT

DNT water

Do you want to use the existing water models? (h/a) n

Is there a water demand at sellitemfr? y or n (h/a) y

Model Water Use at sellitemfr

No file of models found; file created


     BUILD A NEW MODEL for sellitemfr

Input water used as a function of production level

For a max  of 20 points, put in prod. level, up to 100,
not including zero, and water used/generated

Use units of cubic meters/hour or cubic meters/batch for water
(for help with units type 'help')

Prod. level[1]? (h/a) 10

water? 20

Prod. level[2]? (h/a) 50

water? 65

Prod. level[3]? (h/a) 100

water? 105
     105.00:                                                 o
           :                                              ooo
           :                                           ooo
           :                                        ooo
      84.00:                                    oooo
           :                                 ooo
           :                              ooo
           :                           ooo
      63.00:                        ooo
           :                     ooo
           :                   oo
           :                 oo
      42.00:              ooo
           :            oo
           :          oo
w          :       ooo
a     21.00:     oo
t          :ooooo
e          :
r          :
           |---------|---------|---------|---------|---------|
                     20        40        60        80        100
             PRODUCTION LEVEL
Do you want to change this model? y or n n

Is there a water demand at sellitewash? y or n (h/a) n

Is there a water demand at treatmentpond? y or n (h/a) n


water from sellitemfr to sellitewash =  105.00
water from sellitewash to treatmentpond =  105.00

water from treatmentpond to outfall =  105.00

Inflow to network at sellitemfr =  105.00


Do you want another run? y or n (h/a) y


Do you want to change flow connections? y or n (h/a)  n

What production level- 0 to 100? (h/a)
(Previous level was 100)   100

what constituent do you want to model?

The following are available: (h/a)

water

TNT

DNT  TNT

Do you want to use the existing TNT models? (h/a) n

Is TNT generated at sellitemfr? y or n (h/a)  help

Do you want general help or help specific to the question you are on?
Enter 1 or 2:

     1= general
     2=specific
2

Help for question 19. Is constituent generated here
If this is a process at which the oollutant enters the
wastewater stream, and at which there is a pollutant
generation vs. production level model, answer yes
Answering with a 'x' tells the program that there is
neither a pollucant generation nor a treatment model
for this node.

More help?  y or n n

Is TNT generated at sellitemfr? y or n (h/a)  n

Is sellitemfr a treatment process where TNT is removed? (h/a) n

Is TNT generated at sellitewash? y or n (h/a)  y

No file of models found; file created


     BUILD A NEW MODEL for sellitewash

Input TNT generated as a function of production level

For a max. of 20 points, put in prod. level, up to 100,
not including zero, and TNT used/generated

Use units of kg/hour or kg/batch for TNT
(for help with units type 'help')
```

```
Prod  level[1]? (h/a) 10

TNT? 40

Prod. level[2]? (h/a) 60

TNT? 200

Prod. level[3]? (h/a) 1100   80

TNT? 250

Prod. level[4]? (h/a) 100

TNT? 100
```



```
Do you want to change this model? y or n y

     BUILD A NEW MODEL for sellitewash

Input TNT generated as a function of production level

For a max. of 20 points, put in prod. level, up to 100,
not including zero, and TNT used/generated

Use units of kg/hour or kg/batch for TNT
(for help with units type 'help')

Prod. level[1]? (h/a) 10

TNT? 40

Prod. level[2]? (h/a) 60

TNT? 200

Prod. level[3]? (h/a) 80

TNT? 250

Prod. level[4]? (h/a) 100
```

```
TNT? 300
     300.00:                                                 o
           :                                              ooo
           :                                           ooo
           :                                        ooo
     240.00:                                     ooo
           :                                  ooo
           :                               ooo
           :                             oo
     180.00:                          ooo
           :                        oo
           :                      oo
           :                   ooo
     120.00:                 oo
           :               oo
           :            ooo
           :          oo
      60.00:        oo
T          :     ooo
N          :ooooo
T          :
           |---------|---------|---------|---------|---------'
                     20        40        60        80        100
             PRODUCTION LEVEL
Do you want to change this model? y or n n

Is TNT generated at treatmentpond? y or n (h/a) n

Is treatmentpond a treatment process where TNT is removed? (h/a) y

Model Treatment Efficiency at treatmentpond
No model called        treatmentpond in file

     BUILD A NEW MODEL for treatmentpond

Put in removal percent as a function of flow
For a max. of 20 points, put in flow rate and removal

Use units of cubic meters/hour or cubic meters/batch for flow
For help with units type 'help'

Flow rate[1]? (h/a) 50

removal? 98

Flow rate[2]? (h/a) 100

removal? 95

Flow rate[3]? (h/a) 500

removal? 75

Flow rate[4]? (h/a) 0

T      98.00:ooooo
R           :     oooooooo
E           :             oooooooooo
A           :                       oooooooooo
T      78.40:                                 oooooooooo
M           :                                           ooooooo
E           :
N           :
T      58.80:
```

```
          :
E         :
F         :
F    39 20:
I         :
C         :
I         :
E    19 60:
N         :
C         :
Y         :
          |----------|----------|----------|----------|----------|
                    100        200        300        400        500
             FLOW RATE
Do you want to change this model? y or n  n


TNT from sellitemfr to sellitewash =     0.00
Concentration = 0 mg/l

TNT from sellitewash to treatmentpond =  300.00
Concentration = 2857.14 mg/l

TNT from treatmentpond to outfall =   18.00
Concentration = 171.429 mg/l


Do you want another run? y or n (h/a) y

Do you want to change flow connections? y or n (h/a)  y

Wh ch node do you want to change? (h/a) help

Do you want general help or help specific to the question you are on?
Enter 1 or 2:

     1= general
     2=specific
2

Help for question 7:  Which node do you want to change?

You are now in the routine to change flow routing
Enter the name of the node (process) whose outflow you
wish to redirect.

More help?  y or n n

Which node do you want to change? (h/a) treatment        pond

No such node as pond

Which node do you want to change? (h/a) treatmentpond

Flow goes from treatmentpond to outfall

Which node do you wish to eliminate flow to? (h/a) O

Which node do you wish to add flow to? (h/a) sellitemfr

done
```

```
Which node do you wish to add flow to? (h/a) 0

Which node do you want to change? (h/a) 0

Flows changed- model water first to get correct results
What production level- 0 to 100? (h/a)
(Previous level was 100)  100

What constituent do you want to model?

The following are available: (h/a)

water

TNT

DNT  water

Do you want to use the existing water models? (h/a) y

Is there a water demand at sellitemfr? y or n (h/a)  y

Is there a water demand at sellitewash? y or n (h/a)  n

Is there a water demand at treatmentpond? y or n (h/a)  n

Enter fraction of flow from treatmentpond to outfall .8

water from treatmentpond to sellitemfr =   22.00

water from sellitemfr to sellitewash =  110.00

water from sellitewash to treatmentpond =  110.00

water from treatmentpond to outfall =   88.00

Inflow to network at sellitemfr =   88.00

Do you want another run? y or n (h/a) y

Do you want to change flow connections? y or n (h/a)  n

What production level- 0 to 100? (h/a)
(Previous level was 100)  100

What constituent do you want to model?

The following are available: (h/a)

water

TNT

DNT  TNT

Do you want to use the existing TNT models? (h/a) y

Is TNT generated at seilitemfr? y or n (h/a)  x

Is TNT generated at seilitewash? y or n (h/a)  y
Is TNT generated at treatmentpond? y or n (h/a) n

Is treatmentpond a treatment process where TNT is removed? (h/a) y

Enter fraction of flow from treatmentpond to outfall .8


TNT from treatmentpond to sellitemfr =     3.64
Concentration = 165.624 mg/l

TNT from sellitemfr to sellitewash =    3.64
Concentration = 33.1248 mg/l

TNT from sellitewash to treatmentpond =  303.64
Concentration = 2760.4 mg/l

TNT from treatmentpond to outfall =   14.57
Concentration = 165.624 mg/l


Do you want another run? y or n (h/a) y


Do you want to change flow connections? y or n (h/a) n

What production level- 0 to 100? (h/a)
(Previous level was 100)  50

Production level changed- model water first to get correct results
What constituent do you want to model?

The following are available: (h/a)

water

TNT

DNT  water

Do you want to use the existing water models? (h/a) y

Is there a water demand at sellitemfr? y or n (h/a) y

Is there a water demand at sellitewash? y or n (h/a) n

Is there a water demand at treatmentpond? y or n (h/a) n

Enter fraction of flow from treatmentpond to outfall .8


water from treatmentpond to sellitemfr =   13.00

water from sellitemfr to sellitewash =   65.00

water from sellitewash to treatmentpond =   65.00

water from treatmentpond to outfall =   52.00

Inflow to network at sellitemfr =   52.00


Do you want another run? y or n (h/a) y


Do you want to change flow connections? y or n (h/a) n
```

```
What production level- 0 to 100? (h/a)
(Previous level was 50)  50

What constituent do you want to model?

The following are available: (h/a)

water

TNT

DNT  TNT

Do you want to use the existing TNT models? (h/a) y

Is TNT generated at sellitemfr? y or n (h/a)  x

Is TNT generated at sellitewash? y or n (h/a)  y

Is TNT generated at treatmentpond? y or n (h/a)  n

Is treatmentpond a treatment process where TNT is removed? (h/a) u

Enter fraction of flow from treatmentpond to outfall .9

TNT from treatmentpond to sellitemfr =     1.01
Concentration = 78.0065 mg/l

TNT from sellitemfr to sellitewash =    1.01
Concentration = 15.6013 mg/l

TNT from sellitewash to treatmentpond =  169.01
Concentration = 2600.22 mg/l

TNT from treatmentpond to outfall =    4.06
Concentration = 78.0065 mg/l


Do you want another run? y or n (h/a) n
12 >>
```

```
5 )) C cat RESULTS
NEXT RUN

Constituent: water    Production level: 100

water from sellitemfr to sellitewash =  105.00

water from sellitewash to treatmentpond =  105.00

water from treatmentpond to outfall =  105.00

Inflow to network at sellitemfr =  105.00.

NEXT RUN

Constituent: TNT    Production level: 100

TNT from sellitemfr to sellitewash =     0.00
Concentration = 0 mg/l


TNT from sellitewash to treatmentpond =  300.00
Concentration = 2857.14 mg/l


TNT from treatmentpond to outfall =    18.00
Concentration = 171.429 mg/l.

NEXT RUN:

Constituent: water    Production level: 100

water from treatmentpond to sellitemfr =   22.00

water from sellitemfr to sellitewash =  110.00

water from sellitewash to treatmentpond =  110.00

water from treatmentpond to outfall =   88.00

Inflow to network at sellitemfr =   88.00

NEXT RUN

Constituent: TNT    Production level: 100

TNT from treatmentpond to sellitemfr =     3.64
Concentration = 165.624 mg/l


TNT from sellitemfr to sellitewash =     3.64
Concentration = 33.1248 mg/l


TNT from sellitewash to treatmentpond =  303.64
Concentration = 2760.4 mg/l


TNT from treatmentpond to outfall =    14.57
Concentration = 165.624 mg/l.

NEXT RUN

Constituent: water    Production level: 50
```

```
water from treatmentpond to sellitemfr =   13.00

water from sellitemfr to sellitewash =   65.00

water from sellitewash to treatmentpond =   65.00

water from treatmentpond to outfall =   52.00

Inflow to network at sellitemfr =   52.00

NEXT RUN:

Constituent: TNT    Production level: 50

TNT from treatmentpond to sellitemfr =    1.01
Concentration = 78.0065 mg/l


TNT from sellitemfr to sellitewash =    1.01
Concentration = 15.6013 mg/l


TNT from sellitewash to treatmentpond =  169.01
Concentration = 2600.22 mg/l


TNT from treatmentpond to outfall =    4.06
Concentration = 78.0065 mg/l.
```

APPENDIX B: PROGRAM LISTING

```
#include <stdio.h>;

double determ[100][100];
double rhs[100];
double var[100];
int detrmsz;
int matrix[30][30], arysze;

struct a {char aa[20]; };

struct {char ndname[20];
         struct a ndto[6];
         double watrin;
       } node[30];

main()
{

int nvar, i, j, varlbl[100];
int ii, ix, temp, col, row, flag;
int noden, ncnsts, fd;
int tonode[6];
double   flofrac[6], conc;
double qreq, rmval, qtrt;
double flows[100];
float pl, ftemp;
char tquery[4], query[5], ucnstt[20], linebuf[81];
char stemp[5];
struct {char a[20]; } cnsts[20];
/*                                                       */
/*  Bildsys is the main routine. It calls routine Matrxin
    to build a connection matrix for the system.
    Routine chgmtrx allows the user to revise the connection
    matrix. Routine findmdl uses a file of production level
    vs. water demand  models to determine how much water
    is needed at each node(plant process) which requires water.
    Bildsys uses the above information to build a system
    of simultaneous linear equations describing flow through
    the system.  Additional equations are added for each
    node in the system with more than one outflow.

    After the system is built, with the equations described by the
    determinant "determ" and right-side vector "rhs", it is
    solved in the routine "solvsys".
```

```
                                                                    */

/*  variables:                                                  */
/*      nvar        the number of simultaneous equations */
/*                  originally equal to the number of  */
/*                  positive values in the connection matrix */
/*      arysze      the length of the x("from") dimension in the *
/*                  connection matrix                           */
/*      matrix      the connection matrix
        determ      the determinant of the system of simultaneous equations
        rhs         the right side (constants) for the equations
        varlbl      labels variables in the determinanc  */
/*  ***********************************************    */


/*   print out heading */

printf("\n\n\n  Plant Model Program");
printf("\n\n At all prompts which are followed by: (h/a) you can get on-line").
printf("\nhelp by responding with the word 'help'.\n\nAnswer yes no ");
printf("questions with yes, no, y, or n \n\n\n\n");

/* ______________________________________________________ */


/* initialize list of constituents


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  */


setbuf(stdout,NULL);
strcpy(cnsts[0].a,"water");
strcpy(cnsts[1].a,"TNT");
strcpy(cnsts[2].a,"DNT");
ncnsts = 3.


/* {{{{{{{{{{{{{{{{  Initialize watrin  }}}}}}}}}}}}}}}}}}}}}}} */

for (i =0; i <=29; i++)  node[i].watrin = 0.0;
pl = -1,

/*   >>>>>>>>>>>>>> call input routine  <<<<<<<<<<<<<<<<< */

arysze = matrxin(),

/*  >>>>>>>>>>>>>>>>>>> print and modify matrix <<<<<<<<<<<<<<<<<<<  */

for (col = 1, col <= arysze; col++)
    {
            for (row = 1; row <= arysze; row++)
        { if(matrix[row][col] != 0)
            printf("\nFlow goes from %s to %s",node[col].ndname,
            node[row].ndname);
        }
```

```
        printf("\n");
      }

for (col =1, col <= arysze; col++)
     if (matrix[arysze+1][col] != 0)
     printf("\nFlow goes from %s to sink",node[col].ndname);


recycle: printf("\n\nDo you want to change flow connections? y or n (h/a)  ");
scanf("%s",query);

if (strcmp(query,"help") ==0 || strcmp(query,"HELP") == 0 )
        { help(6);
          goto recycle;
        }

if (query[0] == 'y' || query[0] == 'Y')
      { chgmtx();
        printf("\nFlows changed- model water first to get correct results");
      }


ten: printf("\nWhat production level- 0 to 100? (h/a) ");
if(pl > 0.0) printf("\n(Previous level was %-.0f)   ", pl);
scanf("%s",query);

        if (strcmp(query,"help") == 0 || strcmp(query,"HELP") ==0 )
           { help(10);
             goto ten;
           }

sscanf(query,"%f",&ftemp);

                          while (ftemp < 0.0 || ftemp > 100.0)
                        { printf("\nPlease keep production level between 0 and 100"),
                          printf("\n\nProduction level?  ");
                          scanf("%f",&ftemp);
                        }


/*  if the production level is being changed, say so    */

if (ftemp != pl && pl != -1.0)
printf("\nProduction level changed- model water first to get correct results");
pl = ftemp;


i = 99;
while (i >= ncnsts)                /*  read in constituent wanted and
                                       make sure it is available       */
  {
    eleven: printf("\nWhat constituent do you want to model?"),
    printf("\n\nThe following are available: (h/a) ");
         for (i=0; i < ncnsts; i++)
       { printf("\n\n%s  ",cnsts[i].a);
```

```
        }
    scanf("%s",ucnstt);

    if (strcmp(ucnstt,"help") == 0  ||  strcmp(ucnstt,"HELP") ==
       { help(11);
         goto eleven;
       }

         for (i=0; i < ncnsts; i++)
       { if (strcmp(cnsts[i].a,ucnstt) == 0)     break;
       }
    if(i>= ncnsts) printf("\n%s not available",ucnstt);
  }

/*  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    Allow the user to use the existing process models without
    modification.

    &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& */

      twelve: printf("\nDo you want to use the existing %s models? (h/a) ",ucnstt
      scanf("%s",tquery);
      if (tquery[0] == 'Y') tquery[0] = 'y';

      if (strcmp(tquery,"HELP") == 0 || strcmp(tquery,"help") == 0 )
         { help(12);
           goto twelve;
         }


/* ===================================================== */
/*  if modeling water, reset the stored values of water flows to zero */

if(strcmp(ucnstt,"water") == 0) { for (i=0; i <= 29; i++)
                                     node[i].watrin = 0.0;
                                  }

/*  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
       All  the input and bookkeeping is done.  Now build
       the system of simultaneous equation.
   ////////////////////////////////////  ///////////////////////////////// */

nvar=0;
for (i=1,    i<=(arysze+1); i++) {
     for(j=1;    j<=arysze;  j++) {
                  if(matrix[i][j] !=0)  matrix[i][j] = ++nvar;
                                        };
                      ;


/*  array now read into "matrix"                         */
/*  now generate determinant for system of equations */
/* put "1" in determ for inflows      */
/* and "-1" in for outflows           */
for (i=0; i <=99; i++)
```

```
  { for (j=0; j <= 99; j++)  determ[i][j] = 0.0;
    rhs[i] = 0.0;
  }


for ( i=1; i<= arysze; i++)      {
      for (j=1; j<= arysze; j++) {
            if( matrix [i][j] != 0 )       {
                                              determ[i][(matrix[i][j])] = 1.0;
                                              determ[j][(matrix[i][j])] = -1.0;
                                              }; }; };
      for ( j=1; j<= arysze; j++ )
          {
            if( matrix[ (arysze+1) ][j] != 0 )
            determ[j][(matrix[ (arysze+1) ][j]) ] = -1.0;
          }
detrmsz = arysze;

/*   add equations for inflows to system-    */
/*   one equation for each node with a production-level vs. demand function */
/*   for water.  */
/*   call routine "findmdl" to get information on these nodes  */
/*   qreq  is the flow rate demanded at the node   */
/*   add a new row in the determinant for each inflow */
for(i=1; i<=arysze; i++)
    {
      if (strcmp(ucnstt,"water") == 0)
    {
      q13: query[0] = 'n';
      printf("\nIs there a water demand at %s? y or n (h/a)  ",node[i].ndname);
      scanf("%s",query);

      if (strcmp(query,"help") == 0  ||  strcmp(query,"HELP") ==
          { help(13);
            goto q13;
          }


      if(query[0] == 'Y' || query[0] == 'y')


          /*  build new equation if there is inflow   */
         {
           if(tquery[0] != 'y' )
           printf("\nModel Water Use at %s\n",node[i].ndname);

                                   /*  call findmdl with a 1 to allow changes,
                                       with a zero for no changes             */

           if(tquery[0] != 'y')
           qreq = findmdl(node[i].ndname,ucnstt,p1,1);
           else qreq = findmdl(node[i].ndname,ucnstt,p1,0);
           ++detrmsz;
           ++nvar;
           varlbl[nvar] = i;
                for ( j=1;   j<=arusze;   j++ )
            { if( matrix[i][j] != 0 )
              determ[detrmsz][(matrix[i][j]) ] = 1. ;       };
          determ[detrmsz][nvar] = 1.;
          rhs[detrmsz] = qreq;
/*      find the first var. in the node's column      */
ii=1;
while ( matrix[ii][i] == 0 ) ii++;
for ( ix=1;  ix<=detrmsz; ix++ )
   { if( determ[ix][ (matrix[ii][i]) ] == -1.0 ) determ[ix][nvar] = 1.0; };
       }
     } /*  end of water inflow routine.
           Now modify equations for nodes at which pollutants
           are generated.  Change equations from
           "Sum of inflows == Sum of outflows" to
           "Sum of inflows + pollutant generated == sum of outflows"
           This is done by putting -1 * pollut. generated on right
           side of equation.  */


else
   {
     q19  query[0] = 'n';
     printf("\nIs %s generated at %s? y or n (h/a)  ",ucnstt,node[i] ndname);
     scanf("%s",query);

     if(strcmp(query,"help")==0 || strcmp(query,"HELP") == 0)
         { help(19);
           goto q19;
         }

     if(query[0] == 'Y' || query[0] == 'y' )

          {
            if (tquery[0] != 'y')
            qreq = findmdl(node[i].ndname,ucnstt,pl,1);
            else qreq = findmdl(node[i].ndname,ucnstt,ol,0);
            rhs[i] = -1.0 * qreq;
          }



/*   Modify those equations for nodes which are waste treatment processes.
     By multiplying the "-1" values in the equation by the fractional
     removal by the treatment process, the equation is changed from
     "inflow == outflow" to "inflow * removal == outflow"
     This is done only for pollutants, not for water.                    */

else
  {
                              /*
                                  if the user replied to the previous question
                                  with a 'x' assume this is not a
                                  treatment process                          */

    if (query[0] != 'x' && queru[0] != 'X')
```

```
      { q24  printf("\nIs %s a treatment process where %s is removed? (h/a) ",
        node[i].ndname,ucnstt);
        scanf("%s",query);

        if (strcmp(query,"help")==0 || strcmp(query,"HELP") == 0
            { help(24);
              goto q24;
            }
      }
    if (query[0] == 'Y' || query[0] =='y' )
      {
        if (tquery[0] != 'y')
        printf("\nModel Treatment Efficiency at %s",node[i].ndname);

                /*  make sure water has been modeled first and there
                    is a water inflow to this treatment node know   */

                if (node[i].watrin == 0.0)
              { printf("\nWarning- Water flow to process %s is zero- ",
                node[i].ndname);
                printf("\nWere water flows modeled first??");
              }

      /*   if changes in model are allowed, call trtmdl with a 1
           if not , call it with a zero */

        if(tquery[0] != 'y')
        rmval = trtmdl(node[i].ndname,ucnstt,node[i].watrin,1);
        else rmval = trtmdl(node[i].ndname,ucnstt,node[i].watrin,0);

        rmval = 1.0 - (rmval / 100.0);
        for (col=1, col <= nvar, col++)
           {
             if (determ[i][col] > 0.0 )  determ[i][col] =
                                       determ[i][col] * rmval;
           }
       }
   }
  }
   }
/*
       add equations for each node with divided outflows
       telling how much flow goes out each path.
       noden is the node number which has divided outflows
       "tonode" is a matrix of the nodes flow goes to
       "flofrac is a matrix of the fraction of the outflow to the node
                                                                    /

     for(col = 1, col <= arysze; col++)
   { flag = 0;
         for (row =1; row <= arysze+1; row++)
        { if(matrix[row][col] != 0)
             { if (flag != 0)
                     { noden = col,
                       tonode[flag] = row,
                                   q17: if(row == arysze+1)
                                   printf("\nEnter fraction of flow from %s to outfall ",
                                   node[col].ndname);
                                else
                                printf("\nEnter fraction of flow from %s to %s (h/a)  ",
                                node[col].ndname,node[row].ndname);
                                scanf("%s", stemp);

                                if(strcmp(stemp,"help") == 0 || strcmp(stemp,"HELP")==0)
                                   { help(17);
                                     goto q17;
                                   }

                                sscanf(stemp,"%f",flofrac+flag);
                                while (flofrac[flag] < 0.0 || flofrac[flag] > 1.0)
                                    {  printf("\nPlease keep it between 0 and 1.");
                                       printf("    Try again: ");
                                       scanf("%f", flofrac+flag);
                                    }

                              }
                          ++flag;
                      }
            }

   /*  go through equation-adding routine for each node with divided outflow */


   if(flag > 1)
   {

             for (i=1; i<=flag-1; i++)
           {
             ++detrmsz;
             /* temp is the variable number corresponding to
                this flow path (from noden to tonode[])        */
             temp = matrix[(tonode[i])][noden];


             determ[detrmsz][temp] = (1.0 -(flofrac[i]));
                  for(ii=1; ii<=(arysze+1); ii++)
                { if ( (matrix[ii][noden] != 0 ) && ( ii != tonode[i]) )
                  determ[detrmsz][(matrix[ii][noden])] = (-1.0 * (flofrac[i]));
                }
            }
   }
   }
   /*  print out determinant for debug purposes.
   for (i=1; i <= detrmsz; i++)
       { printf("\n");
              for (j=1; j<=detrmsz; j++)
            { printf("%-7.3f", determ[i][j]);
            };
         printf("%10.3f",rhs[i]);
       };                                                         */
   /*
   send determinant to routine solvsys for solution

                                                                    */
solvsys(),
/*  ))))))))))))))))))))))))))))))((((((((((((((((((((((((((((((((((

    print out the results!
    do this both to the standard (terminal) output a::
    to a file called RESULTS.

    ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]][[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[   */

/*     open or create the results file                          */

fd = open("RESULTS", 1);
if (fd < 0 )                fd = creat("RESULTS",0644),
lseek(fd,0L,2);              /*  write the new output at the end of the file
                                 If an overwrite is preferred, change this
                                 statement to lseek(fd,0L,0),                */
strcpy(linebuf,"  NEXT RUN: ");
linebuf[0] = '\n'; linebuf[1] = '\n', linebuf[11] =  \n';
write(fd,linebuf,12);
for(i=0; i<=79; i++)  linebuf[i] = '\0';
sprintf(linebuf,"\nConstituent: %s    Production level  %2.0f",ucnstt,pl);
write(fd,linebuf,81);

/*  print out pipeflows between nodes                          */

nvar = 0,
 for (row=1, row <= arysze, row++)
{     for (col=1, col <= arysze; col++)
   { if (matrix[row][col] != 0 )
          {nvar = nvar + 1,
           printf("\n\n%s from %s to %s = %7.2f",
           ucnstt,node[col].ndname,node[row].ndname,var[matrix[row][col]]),

           for (i=0; i <=79; i++)     linebuf[i] = '\0 ,
           sprintf(linebuf,"\n\n%s from %s to %s = %7.2f".
           ucnstt,node[col].ndname,node[row].ndname,var[matrix[row][col]]);
           write(fd,linebuf,81);


               /*  Save the water inflows to nodes  */
               if (strcmp(ucnstt,"water" ) == 0)
             { node[row].watrin = node[row].watrin + var[matrix[row][col]],
               flows[matrix[row][col]] = var[matrix[row][col]];
             }



/*
          Calculate and print out the concentration of the constituent
          The array "flows" stores the water flow rate in each pipe
          The "else" routine below calculates concentration as flow
          (cubic meters/time) divided by pollutant flow (kg/time) times
          a conversion factor of 1000 to make mg/l.

                                                                              */

               else
              {
               if (flows[matrix[row][col]]==0)  conc = 0;
               else conc = ( var[matrix[row][col]] / flows[matrix[row][col]] )
                         * 1000.0;   /* convert kg/cubic meter to mg/l */
               printf("\nConcentration = %g mg/l",conc);
               for (i=0; i <=79; i++) linebuf[i] = '\0';
               sprintf(linebuf,"\nConcentration = %g mg/l\n",conc),
               write(fd,linebuf,81);
              }


            }
      }
}

/* print out flows to outfall  */

  for (col = 1; col <= arysze; col++)
{ if(matrix[arysze+1][col] != 0 )
      { nvar = nvar + 1;
        printf("\n\n%s from %s to outfall = %7.2f",
        ucnstt,node[col].ndname,var[matrix[arysze+1][col]]);

        for (i=0; i<=79; i++)  linebuf[i] = '\0';
        sprintf(linebuf,"\n\n%s from %s to outfall = %7.2f",
        ucnstt,node[col].ndname,var[matrix[arysze+1][col]]);
        write(fd,linebuf,81);



               if (strcmp(ucnstt,"water" ) == 0)
               flows[matrix[arysze+1][col]] = var[matrix[arysze+1][col]];

/*
          Calculate and print out the concentration of the constituent.
                                                                              */

               else
              {
               if (flows[matrix[arysze+1][col]]==0)  conc = 0;
               else conc = ( var[matrix[arysze+1][col]] /
                flows[matrix[arysze+1][col]] )
                         * 1000.0;   /* convert kg/cubic meter to mg/l */
               printf("\nConcentration = %g mg/l",conc);
               for (i=0; i <=79; i++) linebuf[i] = '\0';
               sprintf(linebuf,"\nConcentration = %g mg/l",conc)
               write(fd,linebuf,81);
              }
      }
}
```

```
/*  print inflows to system where there is a flow requirement  *
printf("\n");

for (i = nvar + 1; i <= detrmsz; i++)
     {
       if (var[i] >= 0.0)
          {
            printf("\nInflow to network at %s = %7.2f\n",
            node[varlbl[i]].ndname,var[i]);

            for(col = 1; col<=79; col++)    linebuf[col] = '\0';
            sprintf(linebuf,"\n\nInflow to network at %s = %7.2f",
            node[varlbl[i]].ndname,var[i]);
            write(fd,linebuf,81);
          }

       else
          {
            var[i] = -1 * var[i];
            printf("\nExcess flow at %s = %7.2f\n",
            node[varlbl[i]].ndname,var[i]);

            for(col = 1; col<=79; col++)    linebuf[col] = '\0';
            sprintf(linebuf,"\n\nExcess flow at %s = %7.2f",
            node[varlbl[i]].ndname,var[i]);
            write(fd,linebuf,81);
          }

     }

write(fd,'\n',1);
close(fd);

/*  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\////////////////////////////////

     Use the "infinitely-abusable goto" to allow rerouting
     of flows and another run of the program.

     ////////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\   *

q18: printf("\n\nDo you want another run? y or n (h/a) ");
scanf("%s",query);

if (strcmp(query,"help") == 0  || strcmp(query,"HELP") == 0)
       { help(18);
         goto q18;
       }

if (query[0] == 'Y' || query[0] == 'y') goto recycle;

                                        } /* end of bildsys */
```

```
#include <stdio.h>;

extern struct a {char aa[20];};
extern struct {char ndname[20];
               struct a ndto[6];
               double watrin;
              } node[30];

extern int matrix[30][30], arysze;

chgmtx()   {

/* ::::::::::::::::::::::::::::::::::: :::::::::::::::::   ::::

   function chgmtx   to make changes in the flow connection
   matrix, allowing the correction of errors or trying
   alternate recycling schemes.

   The important variables are external, as defined elsewhere

   ...   ..............  ............  ............  ......  .....   */

char chgnode[20], elmnode[20], addnode[20];
int i, chgno;

start: printf("\n\nWhich node do you want to change? (h/a) ");
scanf("%s",chgnode);
if(chgnode[0] == '0') return;

if(strcmp(chgnode,"help") == 0  || strcmp(chgnode,"HELP") == 0)
    {
      help(7);
      goto start;
    }

/* find the number of the node to be changed  */

    for (i=1; i <= arysze; i++)
  { if(strcmp(chgnode,node[i].ndname) == 0 )
       { chgno = i;
         break;
       }
  }
if (i > arysze) {printf("\nNo such node as %s",chgnode);
                 goto start;
                }

/*
            print out where flow goes from chgnode

*/
     eight  for (i=1; i <= arysze; i++)
    { if(matrix[i][chgno] != 0)
     printf("\nFlow goes from %s to %s",chgnode,node[i].ndname);
    }
     if(matrix[arysze+1][chgno] != 0)
```

```
      printf("\nFlow goes from %s to outfall",chgnode);

while(elmnode[0] != '0')
  {
    printf("\n\nWhich node do you wish to eliminate flow to? (h/a) ");
    scanf("%s",elmnode);

    if(strcmp(elmnode,"help") == 0  || strcmp(elmnode,"HELP") == 0)
        {
          help(8);
          goto eight;
        }


    if(strcmp(elmnode,"outfall") == 0) { matrix[arysze+1][chgno] = 0;
                                           i = 0;  /* break out of loop */
                                           printf("\ndone");
                                         }
    else
     {
              for (i=1; i <= arysze; i++)
            { if (strcmp(elmnode,node[i].ndname) == 0)
                  { matrix[i][chgno] = 0;
                    printf("\ndone");
                    break;
                  }
            }
     }

if (i > arysze && elmnode[0] != '0')
printf("\nFlow does not go to %s",elmnode);
  }
while(addnode[0] != '0')
  {
    nine  printf("\n\nWhich node do you wish to add flow to? (h/a) ");
    scanf("%s",addnode);

    if(strcmp(addnode,"help") == 0  || strcmp(addnode,"HELP") == 0)
    {
      help(9);
      goto nine;
    }

    if(strcmp(addnode,"outfall") ==0)  { matrix[arysze+1][chgno] = 1;
                                           i = 0;
                                           printf("\ndone");
                                         }
    else
     {
              for (i=1; i <= arysze; i++)
            { if (strcmp(addnode,node[i].ndname) == 0)
                  { matrix[i][chgno] = 1;
                    printf("\ndone");
                    break;
                  }
            }
      }
        if(i > arysze && addnode[0] != '0' )
        printf("\nNo such node as %s",addnode);
  }
elmnode[0] = 'x'; addnode[0] = 'x';
goto start;
                    }




#include <stdio.h>;
/*   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-- +
     routine findmdl to look in a file of existing process
     models and read it if it exists.  Also allows writing
     over existing models.

     Each model in the file has 5 lines: the first is a title lin-
     with an x in the first space; and 4 lines of alternating
     x and y points.  x represents the production level, from
     0 to 100. Y represents the water demanded or mass of waste
     chemical produced.
         Each line must have 80 char. (not including the newline).
     1111111111111111111111111111111.11111111111111111111111111  */



findmdl(wlabel,wmdl,prdlvl,chngs)
char wlabel[20],wmdl[20];
float prdlvl;
int chngs;
{
char linebuf[81], numbuf[9], fletest;
char test[3], label[20], query[10];
int fd, i, j, nread, line, ihi, ilow;
float x[20], y[20];
double qn;

/* initialize x and y */
for(i=0; i <= 19; i++) {x[i] = 0.0; y[i] = 0.0;}

fletest = 'n';
                                     /*  fletest keeps track of what kind
                                         of writing is to be done at end of prog
                                         'r' = replace old model
                                         'n' = write new model in existing file
                                         'o' = use existing model, no overwrite
                                          c' = create new file, write in model


                                    The variable "chngs" is equal to 1 i
                                    chans in the model are to be allowed and
                                    is 0 if no changes allowed
                                                                          */
```

```
fd = open(wmdl,2);
if (fd < 0)
            { printf("\nNo file of models found; file created\n");
              fd = creat(wmdl,0644);
              fletest = 'c';
            }

/*  If file of models exists, print out the names of the models  */

else { lseek(fd,0L,0);


          while((nread = read(fd,linebuf,81)) != 0)
            {
              if (linebuf[0] == 'x')
               {
                                       for(i=0; i <= 19; i++)
                                       label[i] = linebuf[i+1];
                 if(strcmp(label,wlabel) == 0)
                   {
                     fletest = 'o';
                     /* desired model is now found
                        read  it           */

                     for (line = 0; line <= 3; line++)
                       {
                         read(fd,linebuf,81);
                         for (i = 0; i <=4; i++)

                          { for(j = 0; j <= 7; j++)

                              {numbuf[j] = linebuf[(16*i) + j];
                              }
                            numbuf[8] = '\0';
                            sscanf(numbuf,"%8f",x+(line*5 + i));
                            for(j=0; j <= 7; j++)
                              {numbuf[j] = linebuf[(16*i) + j + 8];
                              }
                            numbuf[8] = '\0';
                            sscanf(numbuf,"%8f",y+(line*5 + i));
                          }
                       }
                      break;

                   }
               }
            }

       /*  test if requested model not in file */
       if (fletest != 'o')
         { printf("\nNo model called %20s in file",wlabel);
           goto build;
         }

/*  if no changes are to be made  go to end of program  */
```

```
        if (chngs == 0  &&  fletest == 'o')  goto quit;


        /*  plot out the model; if user wants to change it,
            or if no model was on the file, build a new one   */

        graph(x,y,20,wmdl);
        printf("\n                  PRODUCTION LEVEL");
        printf("\nDo you want to change this model? y or n ");
        scanf("%s",test);
        if(test[0] != 'Y' && test[0] != 'y')  goto quit;
        fletest = 'r';
    }

/*  build new model   */

build: test[0] = 'y';
while (test[0] == 'y')
 {
   for (i=0; i <=19; i++) {x[i] = 0.0; y[i] = 0.0; label[i] = '   }

   printf("\n\n      BUILD A NEW MODEL for %s\n",wlabel);
   strcpy(label,wlabel);
   if(strcmp(wmdl,"water") == 0)
   printf("\nInput water used as a function of production level\n");
   else printf("\nInput %s generated as a function of production level\n",
                 wmdl);
   printf("\nFor a max. of 20 points, put in prod. level, ");
   printf("up to 100,\nnot including zero, and %s used/generated",wmdl);

   if(strcmp(wmdl,"water") == 0)
  printf("\n\nUse units of cubic meters/day or cubic meters/batch for water\n");
   else printf("\n\nUse units of kg/day or kg/batch for %s\n",wmdl);
   printf("(for help with units type 'help')\n");


        q15  for(i=0; i<=19; i++)
      { printf("\nProd. level[%d]? (h/a) ",i+1);
        scanf("%s",query);

        if(strcmp(query,"help") == 0 || strcmp(query,"HELP") == 0)
          { help(15);
            goto q15;
          }

        sscanf(query,"%f",x+i);

        if(x[i] <= 0.0)         {
                                  printf("\nPlease, no negative or zero values");
                                  goto build;
                                }

        printf("\n%s?  ",wmdl);
        scanf("%f",y+i);
        if (x[i] >= 100)   break;   /*  stop at 100% capacity  */
      }
    graph(x,y,20,wmdl);
    printf("\n                PRODUCTION LEVEL");
    printf("\nDo you want to change this model? y or n  ");
    scanf("%s",test);
    if(test[0] == 'Y')    test[0] = 'y';
  }


quit: if (fletest != 'c')
/*   position file for writing       */
/* if replacing old model, move back up 5 lines(405 char) */
   { if (fletest == 'r')   lseek(fd, -405L, 1);

/* if writing new model in file, move to end of file */
     if (fletest == 'n')   lseek(fd, 0L, 2);

/* if creating new file, rewind      */
     if (fletest == 'c')   lseek(fd, 0L, 0);

/*  write out the model,  title line and then 4 data lines.  */

     linebuf[0] = 'x';
     for (i = 0; i <= 19; i++)  linebuf[i+1] = label[i];
     for (i = 21; i <= 79; i++)  linebuf[i] = ' ';
     linebuf[80] = '\n';

     write(fd, linebuf, 81);


         for (line = 0; line <= 3; line++)
       {      for (i=0; i <= 4; i++)
          { sprintf(numbuf,"%8.1e",x[(line*5) + i]);
               for (j=0; j <= 7; j++)
               linebuf[i*16 + j] = numbuf[j];
            sprintf(numbuf,"%8.1e",y[(line*5) + i]);
               for (j=0; j <= 7; j++)
               linebuf[(i*16) +j +8] = numbuf[j];
          }
         linebuf[80] = '\n';
         write(fd,linebuf,81);
       }

   }  /*  end of write routine   */
close(fd);


/* ##############    interpolate value of water demand    ############# */


ilow = 0, ihi = 1;

     if( prdlvl < x[0]) qn = y[0];        /*  Assume that if x less than
                                              min. in graph, y stays at ymin
                                              Don't assume water use goes to
                                              zero at zero production.      */
      else
        {
           while(prdlvl > x[ihi]  && ihi < 20)
               { ++ihi;
                 ++ilow;
               }
           if(ihi >= 20)        /* Don't go off the upper end of the graph */
          {printf("\n%f% production level too high at %s; extend model",
           prdlvl,wlabel);
           goto build;
          }

           qn = y[ilow] +
           (y[ihi] - y[ilow]) * ((prdlvl - x[ilow]) /
           (x[ihi] - x[ilow]));
         }

return(qn);
}



#
/* &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

   Routine Graph to make a graph of a two-dimensional array,
   such as a process model.

   |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| */


graph(x,y,npts,ylabl)
float x[20], y[20];
int npts,
char ylabl[20];
{

float ymax, graphy[53], temp, xmax, q;
int i, ilow, ihi, j;


/*             ***  first find the max. x and y values   ***        */

xmax = 0.0;
ymax = 0.0;
for (i=0; i <= npts-1; i++)
     { if ( y[i] > ymax) ymax = y[i];
       if ( x[i] > xmax) xmax = x[i];
     }

/*  generate 50 points for the graph by interpolation   */
ilow = 0, ihi = 1;
```

```
        for (q= xmax/50.0; q <= xmax; q = q+ (xmax/50.0) )
     {
        j = q * (50.0 / xmax);
        if( q < x[0]) graphy[j] = y[0];          /*  Assume that if x less than
                                                     min. in graph, y stays at ymin
                                                     Don't assume water use goes to
                                                     zero at zero production.      */

        else
          {
              while(q> x[ihi])
                 { ++ihi;
                   ++ilow;
                 }
              graphy[j] = y[ilow] +
              (y[ihi] - y[ilow]) * ((q - x[ilow]) /
              (x[ihi] - x[ilow]));
            }
     }

/*   now print the graph          */

ihi = strlen(ylabl);

        for(i=20; i > 0; i--)
     {
        if(i <= ihi) printf("\n%c",ylabl[ihi-i]);          /*  write the y label */
        else printf("\n ");
        if( i % 4  == 0)  { temp = (ymax / 20) * i;
                            printf("%10.2f!",temp);
                          }
        else printf("           !");

              for(j=1; j<=50; j++)
           { if (graphy[j] < (i+1)*(ymax/20.0) && graphy[j] >= i *(ymax/20.0) )
              printf("o");     /*  print if it is a point */
              else printf(" ");  /* print this if it isn't a point */
           }
     }

/*  print x axis and label */

printf("\n           !");
    for (i=1; i<=5; i++)
    printf("---------!");

printf("\n           ");
    for (i=1; i<=5; i++)      {j = xmax * (i/5.0);
                               printf("       %3d",j); }
return;
}
```

```
# include <stdio h>;
help(n)
int n,
{
/*   help routine for the ammo plant model.   */

int test, i, oldn,
char query[3],

oldn = n,

while(1)
 {
   printf("\nDo you want general help or");
   printf(" help specific to the question you are on? ");
   printf("\nEnter 1 or 2: "),
   printf("\n\n    1= general\n    2=specific\n"),
   scanf("%d",&test);

   if (test == 1)

    {
printf("\n\nAmmunition Plant Process Model\n");
printf("\n  This is a program to model the flow of water and water-borne"),
printf(" pollutants.\nThe following general instructions apply: ");
printf("\n\n1. All process (node) names should contain 20 or fewer"),
printf(" characters of\nany type, but without blanks. ");
printf("  Be careful with spelling!");
printf("\n\n2. Yes or no questions can be answered with: yes, no, y, or n "),
printf("\n\n3. Whenever a prompt repeats itself you can tell it to move on"),
printf("\nby typing '0' (zero). ");
printf("\n\n4  You must always model water flows before pollutant flows "),
printf("because treatment\nmodels use flow rate as a parameter   Any time "),
printf("you change the production\nlevel or re-route flows you should model"),
printf(" water flows again.\n\n");
printf("5. At any prompt where '(h/a)' appears, you can answer 'help' and get"),
printf(" it. ");
printf("\n\nMore details are available in the user manual and details are");
printf(" available\nhere for specific prompts. ");
printf("\n\nHit '1' to continue\n");

scanf("%s",query);

printf("Choose one of the following: ");



      printf("\n\n");
      printf("\n   1  Do you want input to come from a file?");
      printf("\n   2  What is the name of the file?");
      printf("\n   3  What is the name of the next node?");
      printf("\n   4  what node does flow go to?");
      printf("\n   5  Enter node which goes to outfall");
      printf("\n   6  Do you want to change flow connections?");
      printf("\n   7  Which node do you want to change?");
      printf("\n   8  Which node do you wish to eliminate flow to?"),
  printf("\n   9  Which node do you wish to add flow to?");
  printf("\n  10  What production level?");
  printf("\n  11  What constituent do you want to model?");
  printf("\n  12  Do you want to use the existing models?");
  printf("\n  13  Is there a water demand at this node?");
  printf("\n  15  for help with units.");
  printf("\n  17  Enter fraction of flow from a to b");
  printf("\n  18  Do you want another run?");
  printf("\n  19  Is the constituent generated here?");
  printf("\n  24  Is this a treatment process where constituent is");
  printf(" removed?");
  printf("\n  98  for intrepretation of results");
  printf("\n  99  to quit HELP and return to program");

  printf("\n\nWhich one do you want? ");
  printf("(you entered 'help' from no. %d) ",oldn);
  scanf("%d",&n);
}


if (n== 99) return;

else if (n==1)
   { printf("\nHelp for question 1: \n");
     printf("Do you want process configuration input to come from a file?");
     printf("\n\nIf you have a file with all the node names, where flow");
     printf("\ngoes to for each one, and the names of nodes which drain\n");
     printf("to the outfall, you can use it instead of answering\n");
     printf("questions 3 through 5.  This is very useful when running a");
     printf("\nmodel several times.  If you answer yes, the next question");
     printf("\nwill ask you the name of the input file.");
   }

else if (n==2)
   { printf("\nhelp for question 2: What is the name of the file?");
     printf("\n\nIf you told the program that the process configuration");
     printf("\nwas in a file, you now have to tell it the name of the");
     printf(" file.\nIf the file is not in your directory, be sure to");
     printf("\ninclude the path to the file.");
   }

else if (n==3)
   { printf("\n help for question 3: What is the name of the next node?");
     printf("\n\nYou now have to enter the names of all the nodes");
     printf("(processes)\nin your model, one at a time.  Each name can");
     printf("\nhave up to 19 characters, with no blanks.  The program");
     printf("\nis presently dimensioned for up to 30 nodes.");
     printf("\nAfter all nodes are entered, answer this question with");
     printf("\na '0' (zero).");

   }

else if (n==4)
   { printf("\nHelp for question 4: What node does flow go to?");
     printf("\n\nFor the node that you just entered, enter the names");
     printf("\nof the nodes which flow goes to.  You can enter from 0");
```

```
        printf(" to 5\nnodenames; be sure to spell them correctly.  When");
        printf("\nthere are no more to enter, enter a '0' (zero).  After you");
        printf("\nhave entered all the nodes and where flow goes for each");
        printf("\n one, the program will tell you where flow goes from each");
        printf("one\nso you can check it.");
        printf("\nIf it isn't right, you will be allowed to change flow");
        printf("\nconnections at a later prompt.");
        printf("\nDon't worry about flows to the outfall (or otherwise out");
        printf("\nof the the sustem); you will be allowed to tell the");
        printf("\nprogram which nodes go to the outfall later."
      }

  else if (n==5)
        { printf("\nHelp for question 5: Enter node which goes to sink");
          printf("\n\nType in the name of any node (process) which drains to");
          printf("\nthe outfall or otherwise leaves the system.  This promot");
          printf("\nrepeats, so you can have multiple outfalls. Enter '0'");
          printf(" (zero)\nwhen there are no more.");
        }

  else if (n==6)
        { printf("\nHelp for question 6: Do you want to cnange flow ");
          printf("connections?");
          printf("\n\nIf you want to change where flow goes to from any of");
          printf("\nthe nodes, answer yes.  You will enter a routine which");
          printf("\nallows you to do this.");
        }

  else if (n==7)
        {
          printf("\nHelp for question 7:  Which node do you want to change?");
          printf("\n\nYou are now in the routine to change flow routing.");
          printf("\nEnter the name of the node (process) whose outflow you");
          printf("\nwish to redirect.");
        }

  else if (n==8)
        {
          printf("\nHelp for question 8: Which node do you wish to eliminate");
          printf(" flow to?");
          printf("\n\nYou have just been told which nodes, if any, flow goes");
          printf("\nto from the one you want to change.  If you want to stop");
          printf("\nflow from going to any of these, type in the name of");
          printf("\nthat node.  Enter '0' (zero) when you don't want to");
          printf("\nchange any more.");
          printf("\nTo elimate flow to the outfall enter the word 'outfall' ");
        }

  else if (n==9)
        {
          printf("\nHelp for question 9: Which node do you wish to add");
          printf(" flow to?\n\n");
          printf("If you want to add a flow path from this node to some");
          printf("\nother, enter the name of that node.  To add flow to");
          printf("\n the outfall, enter the word 'outfall'.  Enter '0'");
          printf("\n(zero) when you don't want to add any more ");
```

```
    }

else if (n==10)
      { printf("\nHelp for question 10  What production level? ),
        printf("\n\nEnter the production level you want to model  This"),
        printf("\nshould be expressed as a percent of full production,"),
        printf("\nso should be between 0 and 100."),
      }

else if (n==11)
      { printf("\nHelp for question 11. What constituent do you want");
        printf(" to model?\n\n");
        printf("The program will print out a list of the pollutant"),
        printf("\nconstituents for which models exist, and you should").
        printf("\nchoose one of them.  For the first run, and after any"),
        printf("\nchange in flow routing or production level, you must");
        printf("\nmodel water flows first, because the treatment models"),
        printf("\nneed to know the water flow rates.");
      }

else if (n== 12)
      { printf("\nHelp for question 12: Do you want to use existing "),
        printf("models?");
        printf("\n\nIf you answer yes, you will not be allowed to look");
        printf("\nat and modify any of the process models.  If this is");
        printf("\nnot your first time through the program and you are sure").
        printf("\nthat you are satisfied with the models, this will save"),
        printf("\nsome time.  Otherwise, answer no and look at, modify,");
        printf("\nand build models as needed."),
      }

else if (n==13)
      { printf("\nHelp for question 13. Is there a water demand?"),
        printf("\n\nIf this node is one where water is used and where"),
        printf("\na water demand vs. production level model exists (or");
        printf("\nshould exist), answer yes.");
      }

else if (n==15)
      {
        printf("\nHelp for units:\n\n"),
        printf("To get correct concentration calculations, all flows must"),
        printf("\nbe in cubic meters/time and all pollutant generation\n").
        printf("rates must be in kilogram/time.  The time value may be any").
        printf("\nconsistent unit, such as day, or by batch for batch\n").
        printf("processes.  The concentration will be averaged over the"),
        printf("\ntime period chosen.\n\nSome conversion factors.\n\n ),
        printf("Cubic feet x .0283 equals cubic meters."),
        printf("\nLiters x .001 equals cubic meters ");
        printf("\nGallons x  00379 equals cubic meters "),
        printf("\nPounds x .454 equals kilograms.");
        printf("\nCubic feet/sec x 2447 equals cubic meters/day  ),
        printf("\nMillion gal/day x 3786 equals cubic meters/day "),
        printf("\ncubic lightyears/millisecond x 1.02ex-40 equals m3/day").
      }
```

```
else if (n==17)
      { printf("\nHelp for question 17, Enter fraction of flow");
        printf("\nYou must tell the program what fraction of the outflow");
        printf("\nfrom the first node goes to the second.  Enter a number");
        printf("\nbetween zero and one.");
      }

else if (n==18)
      { printf("\nHelp for question 18, Do you want another run");
        printf("\nIf you want to run the program again, type 'y'.");
        printf("\nYou will be allowed to change flow routing, model");
        printf("\nanother constituent, change the production level,");
        printf("\nand all kinds of fun stuff. ");
      }

else if (n==19)
      { printf("\nHelp for question 19, Is constituent generated here");
        printf("\nIf this is a process at which the pollutant enters the");
        printf("\nwastewater stream, and at which there is a pollutant");
        printf("\ngeneration vs. production level model, answer yes");
        printf("\nAnswering with a 'x' tells the program that there is");
        printf("\nneither a pollutant generation nor a treatment model");
        printf("\nfor this node. ");
      }

else if (n==24)
      { printf("\nHelp for question 24, Is this a process where");
        printf("\n the the constituent is removed");
        printf("\nIf this is a process where the pollutant is removed");
        printf("\nfrom the wastewater stream, such as a treatment process");
        printf("\nand where treatment efficiency vs. flow rate is to be");
        printf("\nmodeled, answer yes.");
      }

else if (n==98)
      {
        printf("\nHelp for interpretation of results.\n\n");
        printf("There are four different kinds of output lines:");
        printf("\n\n'<constituent> from <node> to <node> = ___'");
        printf("\n\nThis line tells you the flow of the constituent");
        printf("\n(or water) in the pipe between the two nodes   Flow");
        printf("\nrates are the same as used in the water use/pollutant");
        printf("\ngeneration graphs.");
        printf("\n\n'Concentration = ___'\n");
        printf("\nThis line tells you the concentration of the constituent");
        printf("\nin the pipe when modeling pollutant flows.")
        printf("\n\n'Inflow to network at <node> = ___'\n");
        printf("\nThis line appears when modeling water flows and tells ");
        printf("\nyou the rate at which water enters the system from");
        printf("\nwhatever the source is (well, intake from river,");
        printf("\ntreatment plant, etc.).");
        printf("\n\n'Excess flow at <node> = ___'\-");
        printf("\nThis line appears when there is more water flowing into");
        printf("\nthe node from upstream nodes than is needed, as");
        printf("\ndetermined by the water demand graph.");
        printf("\nThis excess water leaves the system at this point and");
```

```
          printf("\ncould be considered as flow to the outfall.");
          printf("\nThe presence of excess flow means that you are routing ");
          printf("\ntoo much water into the node.");
        }

    else
        { printf("\nSorry, you picked an option which isn't available ");
        }

    printf("\n\nMore help?  y or n ");
    scanf("%s",query);
    if (query[0] != 'Y' && queru[0] != 'y')    return;
  }
}



#include <stdio.h>;
extern int matrix[30][30];


/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

   subroutine matrxin  to build the connection matrix. Read in
   node labels, nodes being plant processes or treatment
   processes. Nodes representing sources to the system (withdrawals
   from the river etc.) are added automatically.

   variables:

   ndname-  an array of the node names. The node number is assigned
            as the position in the array.

   ndto-    an array of the node numbers vs. the node numbers flow goes
            to.


   mtxsze-  the number of columns in the matrix. There is one more
            row than columns to accomodate the sink.


   Modified to allow input from a file.
   :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: */

struct a { char aa[20]; };

struct { char ndname[20];
         struct a ndto[6];
         double watrin;
       } node[30];
```

```
matrxin()                                {

int z, i, iy, j, ndno, mtxsze;
char tosink[20], test[4];
char filenm[20];
FILE *fp, *fopen();

one: printf("\nDo you want process configuration input ");
printf("to come from a file? (h/a) ");
scanf("%s",test);

if (test[0] == 'H' || test[0] == 'h')    { help(1);
                                            goto one;
                                          }

if (test[0] != 'Y'  &&  test[0] != 'y')
 {

printf("\nProcess configuration entry routine: put in the name");
printf("\nand where flow goes to for each node.\n\n");


for (z = 1; z <= 30; z++)
     { three: printf("\nWhat is the name of the next node?");
       printf("   Hit zero for no more  (h/a) ");
       scanf("%s",node[z].ndname);
       printf("%s",node[z].ndname);
       if (node[z].ndname[0] == '0') break;
       if (strcmp(node[z].ndname,"help") == 0 ||
       strcmp(node[z].ndname,"HELP") == 0 )
                     { help(3);
                       goto three;
                     }

           for (i = 1; i <= 5; i++)
         { four: printf("\nWhat node does flow go to from %s? (h/a)  ",
           node[z].ndname);
           scanf("%s",node[z].ndto[i].aa);
           printf("%s",node[z].ndto[i].aa);
           if (node[z].ndto[i].aa[0] == '0')     break;
           if (strcmp(node[z].ndto[i].aa,"help") == 0 ||
           strcmp(node[z].ndto[i].aa,"HELP") == 0 )
                     { help(4);
                       goto four;
                     }
         }
      }

 }


else    {
          two: printf("\nWhat is the name of the file? (h/a) ");
          scanf("%s",filenm);
          if (strcmp(filenm,"help") == 0 || strcmp(filenm,"HELP") == 0 )
                { help(2);
                  goto two;
                }
          while ((fp = fopen(filenm,"r")) == 0)
           { printf("\nFile %s not found. Please try again (h/a)  ",filenm);
             scanf("%s",filenm);
             if (strcmp(filenm,"help") == 0 || strcmp(filenm,"HELP") == 0 )
                { help(2);
                  goto two;
                }
           }


          for (z=1;z <= 30; z++)
           { fscanf(fp,"%s",node[z].ndname);
             if (node[z].ndname[0] == '0') break;

                  for (i=1; i <= 5; i++)
                { fscanf(fp,"%s",node[z].ndto[i].aa);
                  if (node[z].ndto[i].aa[0] == '0')     break;
                }
           }

        }
mtxsze = (z-1);


/*   The node names and where flow goes to have now been read in.
     Now construct the matrix.  The column number represents
     the node flow comes from and the row number represents the
     node flow goes to.


     for (i=1; i <=mtxsze; i++)
  {     for (j=1; j <= 6; j++)
      { if (node[i].ndto[j].aa[0] == '0' )     break;

        /*  look up the node with a node no. equal to ndto
            (find the node no. flow goes to, give its name)  */

            for (iy = 1; iy <= mtxsze; iy++)
          { if (strcmp(node[i].ndto[j].aa,node[iy].ndname) == 0 )
             { matrix[iy][i] = 1; break;
             }

          }
        if (iy > mtxsze)
             {
               printf("no node called %s",node[i].ndto[j].aa);
               printf("\nInput error occurred- program aborted\n\nYou lose\n\n");
                exit();
             }
        }
    }

/*  add flows to sink or outfall   */
if(test[0] != 'Y'  &&  test[0] != 'y')
 {

        for ( z=1; z <= mtxsze; z++)
     { five: printf("\nEnter node which goes to sink (h/a) ");
        scanf("%s",tosink);
        if (tosink[0] == '0') break;

        if(strcmp(tosink,"help") == 0  ||  strcmp(tosink,"HELP") == 0)
              { help(5);
                goto five;
              }

            for (i=1; i<=mtxsze; i++)
          { if (strcmp(tosink,node[i].ndname) == 0 )
               { matrix[mtxsze+1][i] = 1;
                 break;
               }
          }
        if (i > mtxsze) printf("\nNo such node as %s",tosink);
     }

 }

else   { for (z=1; z <= mtxsze; z++)
           { fscanf(fp,"%s",tosink);
             if (tosink[0] == '0')   break;
               for (i=1; i <= mtxsze; i++)
             { if (strcmp(tosink,node[i].ndname) == 0 )
                 { matrix[mtxsze+1][i] = 1;
                   break;
                 }
             }
           }
         fclose(fp);
       }

/*  print out labelled matrix

printf("\n              %-20s",node[1].ndname);
for (i=2; i <= mtxsze; i++) { printf("%-15s",node[i].ndname); }
for (i=1; i <= mtxsze; i++)
          { printf("\n%-15s",node[i].ndname);
              for (j=1; j <= mtxsze; j++)
            { printf("%-15d",matrix[i][j]);
            }
          }

printf("\nSink /outfall  ");
for (i=1; i <= mtxsze; i++)
printf("%-15d",matrix[mtxsze+1][i]);                     */
```

```
return(mtxsze);
                                        /* end matrxin */



#include <stdio.h>;
extern double determ[100][100];
extern int detrmsz;
extern double rhs[100], var[100];
solvsys()                             {
int i, iswap, ii, row, col;
double swapo, mltplr;
double count;


/*     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        make sure the diagonal of the determinant has no zeroes-
        this is necessary to make the solution routine work.
         First go down  hrough the rows substituting them to
       put values in the diagonal.  If this does not fill in
       the entire diagonal, continue by adding rows.
     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^        */

count = 0.0;
for (i=1; i <= detrmsz; i++)
    { if (determ[i][i] == 0.0)
          { count = 1.0;
            for (ii = 1+i; ii <= detrmsz; ii++)
              { if (determ[ii][i] != 0.0)
                  { for (iswap = 1; iswap <= detrmsz, iswap++)
                      { swapo = determ[i][iswap];
                        determ[i][iswap] = determ[ii][iswap],
                        determ[ii][iswap] = swapo;
                      }
                     swapo = rhs[i];
                     rhs[i] = rhs[ii];
                     rhs[ii] = swapo;
                     count = 0.0;
                     break;
                  }
              }
          }
      }

/*   for diagonal elements not changed from zero by substitution,
     go back through and add an equation with a non-zero element
     in the appropriate column.
                                                                 */

if (count != 0.0)
    {     for (i=1; i <= detrmsz; i++)
       { if (determ[i][i] == 0.0)
            { for (ii = 1; ii <=detrmsz; ii++)
                if (determ[ii][i] != 0.0 )
```

```
                    { for (iswap= 1; iswap <=detrmsz; iswap++)
                      determ[i][iswap] = determ[i][iswap] + determ[ii][iswap];
                      rhs[i] = rhs[i] + rhs[ii];
                      break;
                    }
                }

           }
        }



/*     Gaussian Elimination Routine (since the iterative one did'nt
       converge).
       First eliminate variables below the diagonal                    */


        for (col = 1; col <= detrmsz; col++)
      {        for(row = col + 1; row <= detrmsz; row++)
           { if(determ[row][col] == 0.0) continue;
                      if(determ[col][col] == 0.0)
                    { for(iswap = 1; iswap <= detrmsz; iswap++)
                      determ[col][iswap] = determ[col][iswap] +
                      determ[row][iswap];
                      rhs[col] = rhs[col] + rhs[row];

                    }
             mltplr = determ[row][col] / determ[col][col];
                 for(i=1; i<= detrmsz; i++)
               { determ[row][i] = determ[row][i] - (mltplr * determ[col][i]);
               }
             rhs[row] = rhs[row] - (mltplr * rhs[col]);

           }
/*   debug print routine
       printf("\n\nDEBUG PRINT\n");
         for(row = 1; row <= detrmsz; row++)
      {
         printf("\n");
            for (iswap = 1; iswap <= detrmsz; iswap++)
          { printf("%-7.3f",determ[row][iswap]); }
         printf("   %-7.3f",rhs[row]);
      }                                                       */

      }

/*    Now solve for the variables, working up the diagonal
      from the bottom right.

       for(row = detrmsz; row >= 1; row--)
      {  var[row] = 0.0;
            for(col = row+1; col <= detrmsz; col++)
          { var[row] = var[row] - (determ[row][col] * var[col]);
          }
        var[row] = (var[row] + rhs[row]) / determ[row][row];
```

```
  }

/*   print out variable values.  (THE SOLUTION!)
      for (i=1; i<=detrmsz; i++)
      printf("\nvariable %d = %7.3f", i, var[i]);
                                                                  */

return; }
#include <stdio.h>;
/*  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    routine findmdl to look in a file of existing process
    models and read it if it exists.  Also allows writing
    over existing models.

    Each model in the file has 5 lines: the first is a title line
    with an x in the first space; and 4 lines of alternating
    x and y points.  x represents the production level, from
    0 to 100. Y represents the water demanded or mass of waste
    chemical produced.
        Each line must have 80 char. (not including the newline).
    11111111111111111111111111111111111111111111111111111111111111111  */


trtmdl(wlabel,wmdl,qin,chgs)
char wlabel[20], wmdl[20];
float qin;
int chgs;
{
char linebuf[81], numbuf[9], label[20], fletest;
char test[3],
char query[9];
int fd, i, j, nread, line, ihi, ilow;
float x[20], y[20];
double qn;

/* initialize x and y */
for(i=0; i <= 19; i++) {x[i] = 0.0; y[i] = 0.0;}

fletest = 'n';
                                /*  fletest keeps track of what kind
                                    of writing is to be done at end of prog
                                    '-' = replace old model
                                      ' = write new model in existing file
                                     ' = use existing model, no overwrite
                                     ' = create new file, write in model

                 The variable chgs  is passed as a 1 if
                changes to models are to be allowed; a 0 if not.
                                                                        */


fd = open(wmdl,2);
if (fd < 0)
```

```
                { printf("\nNo file of models found; file created\n");
                  fd = creat(wmdl,0644);
                  fletest = 'c';
                }

/*  If file of models exists, print out the names of the models  */

else { lseek(fd,0L,0);


           while((nread = read(fd,linebuf,81)) != 0)
                {
                  if (linebuf[0] == 'x')
                   {
                                            for(i=0; i <= 19; i++)
                                            label[i] = linebuf[i+1];
                     if(strcmp(label,wlabel) == 0)
                       {
                         fletest = 'o';
                         /* desired model is now found
                            read  it          */

                         for (line = 0; line <= 3; line++)
                           {
                             read(fd,linebuf,81);
                             for (i = 0; i <=4; i++)

                              { for(j = 0; j <= 7; j++)

                                   {numbuf[j] = linebuf[(16*i) + j];
                                   }
                                numbuf[8] = '\0';
                                sscanf(numbuf,"%8f",x+(line*5 + i));
                                for(j=0; j <= 7; j++)
                                   {numbuf[j] = linebuf[(16*i) + j + 8];
                                   }
                                numbuf[8] = '\0';
                                sscanf(numbuf,"%8f",y+(line*5 + i));
                              }
                           }
                          break;

                       }
                   }
                }

        /*  test if requested model not in file */
        if (fletest != 'o')
          { printf("\nNo model called %20s in file",wlabel);
            goto build;
          }
        /*  if no changes are to be allowed  go to end of routine  */

        if (chgs == 0  && fletest == 'o')  goto quit;

        /*  plot out the model; if user wants to change it,
            or if no model was on the file, build a new one.   */

        graph(x,y,20,"TREATMENT EFFICIENCY");
        printf("\n                  FLOW   RATE");
        printf("\nDo you want to change this model? y or n ");
        scanf("%s",test);
        if(test[0] != 'Y' && test[0] != 'y')  goto quit;
        fletest = 'r';
    }

/*  build new model   */

build: test[0] = 'y';
while (test[0] == 'y')
 {
    for (i=0; i <=19; i++) {x[i] = 0.0; y[i] = 0.0; }
    printf("\n\n      BUILD A NEW MODEL for %s\n",wlabel);
    for(i=0; i <= 19; i++)     label[i] = ' ';
    /*  printf("\nPut in process name  ");
    scanf("%20s",label);     */
    printf("\nPut in removal percent as a function of flow");
    printf("\nFor a max. of 20 points, put in flow rate ");
    printf("and % removal");
    printf("\n\nUse units of cubic meters/day or cubic meters/batch for flow\n");
    printf("For help with units type 'help'\n");

        q15: for(i=0; i<=19; i++)
      { printf("\nFlow rate[%d]? (h/a) ",i+1);
        scanf("%s",query);

        if(strcmp(query,"help") == 0 || strcmp(query,"HELP") == 0)
          { help(15);
            goto q15;
          }
        sscanf(query,"%f",x+i);
        if (x[0] <= 0.0)       { printf("\nPlease, no zero or negative values");
                                 goto build;
                               }
        if (x[i] <= 0.0)  break;


        printf("\n% removal?  ");
        scanf("%f",y+i);
      }
    graph(x,y,20,"TREATMENT EFFICIENCY");
    printf("\n               FLOW RATE");
    printf("\nDo you want to change this model? y or n  ");
    scanf("%s",test);
    if(test[0] == 'Y')    test[0] = 'y';
 }


quit: if (fletest != 'o')
/*   position file for writing     */
/* if replacing old model, move back up 5 lines(405 char) */
   { if (fletest == 'r')  lseek(fd, -405L, 1);
```

```
/* if writing new model in file, move to end of file */
     if (fletest == 'n')  lseek(fd, 0L, 2);

/* if creating new file, rewind     */
     if (fletest == 'c')  lseek(fd, 0L, 0);

 *  write out the model,  title line and then 4 data lines.  */

     linebuf[0] = 'x',
     for (i = 0; i <= 19; i++)  linebuf[i+1] = wlabel[i];
     for (i = 21, i <= 79; i++)  linebuf[i] = ' ';
     linebuf[80] = '\n',

     write(fd, linebuf, 81);


         for (line = 0; line <= 3; line++)
       {     for (i=0; i <= 4; i++)
          { sprintf(numbuf, "%8.1e", x[(line*5) + i]);
                for (j=0; j <= 7; j++)
                linebuf[i*16 + j] = numbuf[j];
            sprintf(numbuf, "%8.1e", y[(line*5) + i]);
                for (j=0; j <= 7; j++)
                linebuf[(i*16) +j +8] = numbuf[j];
          }
         linebuf[80] = '\n';
         write(fd,linebuf,81);
       }

    }  /*  end of write routine  */
close(fd);


/* #############   interpolate value of water demand   ############# */


ilow = 0, ihi = 1,

     if( qin < x[0]) qn = y[0];       /*  Assume that if x less than
                                          min. in graph, y stays at ymin
                                          Don't assume water use goes to
                                          zero at zero production.      */

     else
       {
          while(qin > x[ihi]  && ihi < 20)
             { ++ihi;
               ++ilow;
             }
          if (ihi >= 20)
         {printf("\nFlow rate of %f too high at %s, extend model.
          qin,wlabel);
          goto build;
         }
          qn = y[ilow] +

         (y[ihi] - y[ilow]) * ((qin - x[ilow]) /
         (x[ihi] - x[ilow]));
       }

return(qn),
}
```

CERL DISTRIBUTION

Chief of Engineers
ATTN: Tech Monitor
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-CCP
ATTN: DAEN-CW
ATTN: DAEN-CWE
ATTN: DAEN-CWR-W
ATTN: DAEN-CWO
ATTN: DAEN-CWP
ATTN: DAEN-EC
ATTN: DAEN-ECC
ATTN: DAEN-ECE
ATTN: DAEN-ZCF
ATTN: DAEN-ECB
ATTN: DAEN-RD
ATTN: DAEN-RDC
ATTN: DAEN-RDM
ATTN: DAEN-RM
ATTN: DAEN-ZCZ
ATTN: DAEN-ZCE
ATTN: DAEN-ZCI
ATTN: DAEN-ZCM

FESA, ATTN: Library 22060

FESA, ATTN: DET III 79906

US Army Engineer Districts
ATTN: Library
Alaska 99501
Al Batin 09616
Albuquerque 87103
Baltimore 21203
Buffalo 14207
Charleston 29402
Chicago 60604
Detroit 48231
Far East 96301
Fort Worth 76102
Galveston 77550
Huntington 25721
Jacksonville 32232
Japan 96343
Kansas City 64106
Little Rock 72203
Los Angeles 90053
Louisville 40201
Memphis 38103
Mobile 36628
Nashville 37202
New England 02154
New Orleans 70160
New York 10007
Norfolk 23510
Omaha 68102
Philadelphia 19106
Pittsburgh 15222
Portland 97208
Riyadh 09038
Rock Island 61201
Sacramento 95814
San Francisco 94105
Savannah 31402
Seattle 98124
St. Louis 63101
St. Paul 55101
Tulsa 74102
Vicksburg 39180
Walla Walla 99362
Wilmington 28401

US Army Engineer Divisions
ATTN: Library
Europe 09757
Huntsville 35807
Lower Mississippi Valley 39180
Middle East 09038
Middle East (Rear) 22601
Missouri River 68101
North Atlantic 10007
North Central 60605
North Pacific 97208
Ohio River 45201
Pacific Ocean 96858
South Atlantic 30303
South Pacific 94111
Southwestern 75202

US Army Europe
HQ, 7th Army Training Command 09114
ATTN: AETTG-DEH (5)
HQ, 7th Army ODCS/Engr. 09403
ATTN: AEAEN-EH (4)
V. Corps 09079
ATTN: AETVDEH (5)
VII. Corps 09154
ATTN: AETSDEH (5)
21st Support Command 09325
ATTN: AEREH (5)
Berlin 09742
ATTN: AEBA-EN (2)
Southern European Task Force 09168
ATTN: AESE-ENG (3)
Installation Support Activity 09403
ATTN: AEUES-RP

8th USA, Korea
ATTN: EAFE (8) 96301
ATTN: EAFE-Y 96358
ATTN: EAFE-ID 96224
ATTN: EAFE-AM 96208

8th USA, Korea
ATTN: EAFE-H 96271
ATTN: EAFE-P 96259
ATTN: EAFE-T 96212

ROK/US Combined Forces Command 96301
ATTN: EUSA-HHC-CFC/Engr

USA Japan (USARJ)
Ch, FE Div, AJEN-FE 96343
Fac Engr (Honshu) 96343
Fac Engr (Okinawa) 96331

Rocky Mt. Area 80903

Area Engineer, AEDC-Area Office
Arnold Air Force Station, TN 37389

Western Area Office, CE
Vanderberg AFB, CA 93437

416th Engineer Command 60623
ATTN: Facilities Engineer

US Military Academy 10996
ATTN: Facilities Engineer
ATTN: Dept of Geography & Computer Science
ATTN: DSCPER/MAEN-A

Engr. Studies Center 20315
ATTN: Library

AMMRC, ATTN: DRXMR-WE 02172

USA ARRCOM 61299
ATTN: DRCIS-RI-I
ATTN: DRSAR-IS

DARCOM - Dir., Inst., & Svcs.
ATTN: Facilities Engineer
ARRADCOM 07801
Aberdeen Proving Ground 21005
Army Matls. and Mechanics Res. Ctr.
Corpus Christi Army Depot 78419
Harry Diamond Laboratories 20783
Dugway Proving Ground 84022
Jefferson Proving Ground 47250
Fort Monmouth 07703
Letterkenny Army Depot 17201
Natick R&D Ctr. 01760
New Cumberland Army Depot 17070
Pueblo Army Depot 81001
Red River Army Depot 75501
Redstone Arsenal 35809
Rock Island Arsenal 61299
Savanna Army Depot 61074
Sharpe Army Depot 95331
Seneca Army Depot 14541
Tobyhanna Army Depot 18466
Tooele Army Depot 84074
Watervliet Arsenal 12189
Yuma Proving Ground 85364
White Sands Missile Range 88002

DLA ATTN: DLA-WI 22314

FORSCOM
FORSCOM Engineer, ATTN: AFEN-FE
ATTN: Facilities Engineer
Fort Buchanan 00934
Fort Bragg 28307
Fort Campbell 42223
Fort Carson 80913
Fort Devens 01433
Fort Drum 13601
Fort Hood 76544
Fort Indiantown Gap 17003
Fort Irwin 92311
Fort Sam Houston 78234
Fort Lewis 98433
Fort McCoy 54656
Fort McPherson 30330
Fort George G. Meade 20755
Fort Ord 93941
Fort Polk 71459
Fort Richardson 99505
Fort Riley 66442
Presidio of San Francisco 94129
Fort Sheridan 60037
Fort Stewart 31313
Fort Wainwright 99703
Vancouver Bks. 98660

HSC
ATTN: HSLO-F 78234
ATTN: Facilities Engineer
Fitzsimons AMC 80240
Walter Reed AMC 20012

INSCOM - Ch, Instl. Div.
ATTN: Facilities Engineer
Arlington Hall Station (2) 22212
Vint Hill Farms Station 22186

MDW
ATTN: Facilities Engineer
Cameron Station 22314
Fort Lesley J. McNair 20319
Fort Myer 22211

MTMC
ATTN: MTMC-SA 20315
ATTN: Facilities Engineer
Oakland Army Base 94626
Bayonne MOT 07002
Sunny Point MOT 28461

NARADCOM, ATTN: DRDNA-F 071160

TARCOM, Fac. Div. 48090

TRADOC
HQ, TRADOC, ATTN: ATEN-FE
ATTN: Facilities Engineer
Fort Belvoir 22060
Fort Benning 31905
Fort Bliss 79916
Carlisle Barracks 17013
Fort Chaffee 72902
Fort Dix 08640
Fort Eustis 23604
Fort Gordon 30905
Fort Hamilton 11252
Fort Benjamin Harrison 46216
Fort Jackson 29207
Fort Knox 40121
Fort Leavenworth 66027
Fort Lee 23801
Fort McClellan 36205
Fort Monroe 23651
Fort Rucker 36362
Fort Sill 73503
Fort Leonard Wood 65473

TSARCOM, ATTN: STSAS-F 63120

USACC
ATTN: Facilities Engineer
Fort Huachuca 85613 (2)
Fort Ritchie 21719

WESTCOM
ATTN: Facilities Engineer
Fort Shafter 96858
ATTN: APEN-IM

SHAPE 09055
ATTN: Survivability Section, CCB-OPS
Infrastructure Branch, LANDA

HQ USEUCOM 09128
ATTN: ECJ 4/7-LOE

Fort Belvoir, VA 22060
ATTN: ATZA-DTE-EM
ATTN: ATZA-DTE-SW
ATTN: ATZA-FE
ATTN: Engr. Library
ATTN: Canadian Liaison Office (2)
ATTN: IWR Library

Cold Regions Research Engineering Lab 03755
ATTN: Library

ETL, ATTN: Library 22060

Waterways Experiment Station 39180
ATTN: Library

HQ, XVIII Airborne Corps and 28307
Ft. Bragg
ATTN: AFZA-FE-EE

Chanute AFB, IL 61868
3345 CES/DE, Stop 27

Norton AFB 92409
ATTN: AFRCE-MX/DEE

Tyndall AFB, FL 32403
AFESC/Engineering & Service Lab

NAFEC
ATTN: RDT&E Liaison Office
Atlantic Division 23511
Chesapeake Division 20374
Southern Division 29411
Pacific Division 96860
Northern Division 19112
Western Division 64066
ATTN: Sr. Tech. FAC-03T 22332
ATTN: Asst. CDR R&D, FAC-03 22332

NCEL 93041
ATTN: Library (Code L08A)

Defense Technical Info. Center 22314
ATTN: DDA (12)

Engineering Societies Library 10017
New York, NY

National Guard Bureau 20310
Installation Division

US Government Printing Office 22304
Receiving Section/Depository Copies (2)

268
1-83

Users guide : Simulation model for ammunition plants : prediction of wastewater characteristics and impact reuse-recycle / by Steven Railsback ... (et al.) - Champaign, IL ; Construction Engineering Research Laboratory ; available from NTIS, 1983.
65 page (Technical Report / Construction Engineering Research Laboratory ; N-152)

1. Ammunition-manufacture-environmental aspects. 2. Water-pollution-computer programs. I. Railsback, Steven F. II. Messenger, Manette. III. Webster, Ronald Dwight. IV. Bandy, John T. V. Series : Technical Report (Construction Engineering Research Laboratory) ; N-152.

ATE
LMED
8